VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH A IMPLEMENTACE NAPOJENÍ INFORMAČNÍHO SYSTÉMU SPOLEČNOSTI NA EKONOMICKÝ SOFTWARE

DESIGN AND IMPLEMENTATION OF COMPANY INFORMATION SYSTEM CONNECTION TO ECONOMIC SOFTWARE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. ONDŘEJ MACHALA
AUTHOR

VEDOUCÍ PRÁCE — Ing. BERNARD NEUWIRTH, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Záměrem práce je analýza, návrh a implementace datového mostu mezi informačním systémem společnosti Sledovanitv.cz, s.r.o. a ekonomickým systémem. V první části práce jsou stanoveny cíle projektu. Teoretická část poskytuje nutný základ pro realizační činnosti a kromě teoretického pozadí informačních systémů představuje a popisuje technologie, na kterých bude realizována implementace. Na základě analýzy současného stavu jsou stanoveny potřeby a požadavky budoucího rozšíření. V praktické části práce je představen návrh řešení, způsob a postup implementace. V závěrečné části autor přináší zhodnocení praktických i ekonomických přínosů projektu.

Klíčová slova

Informační systém, životní cyklus IS, metody a strategie zavádění IS, metoda HOS, PHP 5, MariaDB, Nette framework 2, architektura MVC, architektura MVP, ekonomický systém FlexiBee, HTTP, REST-API.

Abstract

The aim of this thesis is the analysis, design and implementation of a data bridge between the information system of the company Sledovanitv.cz, s.r.o. and the economic system. The first part describes the objectives of the project. The theoretical part provides the necessary foundation for the implementation and, in addition to the theoretical background of information systems as a whole it presents and describes the technologies used for the execution of the implementation. The needs and requirements for future expansion are determined based on the analysis of the current situation. The practical part introduces a proposed solution, the method and procedure of implementation. In the final part, the author gives an assessment of the practical and economic benefits of the project.

Keywords

Information system, life cycle of IS, methods and strategies for implementing IS, HOS 8 method, PHP 5, MariaDB, Nette framework 2, MVC architecture, MVP architecture, eonomic system FlexiBee, HTTP, REST-API.

BIBLIOGRAFICKÁ CITACE

MACHALA, O. Návrh a implementace napojení informačního systému společnosti na ekonomický software. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2014. 84 s. Vedoucí diplomové práce Ing. Bernard Neuwirth, Ph.D..

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským, ve znění pozdějších předpisů).

V Brně dne 25. května 2014

……
Bc. Ondřej Machala

PODĚKOVÁNÍ

Tímto děkuji vedoucímu diplomové práce panu Ing. Bernardu Neuwirthovi, Ph.D. za cenné rady, připomínky a metodické vedení práce. Velké dík patří společnosti Sledovanitv, s.r.o. za umožnění realizace práce a zde především hlavnímu vývojáři - Luďkovi Svobodovi za trpělivost a podporu.

OBSAH

ÚVOD

Najít obor lidské činnosti, do kterého doposud informační technologie nijak nepronikly, začíná být takřka nemožné. Jsme svědky přeměny ze stavu, kdy firmy informační technologie používaly pouze pro podporu některých svých činností do stavu, kdy na informační infrastruktuře zcela závisí jejich podnikání. Vítejte ve světě informačních systémů.

Schopnost automatizovat činnosti od těch rutinních až po složené a komplexní úlohy firmám přinesla obrovskou konkurenční výhodu, která v posledních letech přispěla k nebývalému rozvoji těchto projektů, které jsou dnes označovány jako startupy.

Úspěch technologických firem dnes kromě know-how zásadním způsobem závisí také na efektivnosti, s jakou umí technologie využívat a s jakou dokáží vytěžovat data. Podstatou dosažení úspěchu u těchto firem je správný výběr a využití existujících technologií a kvalitní vlastní vývoj, který výstupu dodá přidanou hodnotu.

Tímto se dostáváme k předmětu této práce, kterým je analýza stávajícího stavu informačního systému společnosti a nalezení nedostatků, přičemž výstupem bude nejen nalezení optimálního řešení, ale uvedení řešení do praxe a reálné zhodnocení přínosů. Za kritické považuji počáteční fáze, kde bylo nutné analyticky určit, které technologie a jakým způsobem budou využívány, co přesně bude výstupem a jakým způsobem bude výstup vlastní tvorbou realizován v předmětné organizaci.

Tato diplomová práce je realizována v prostředí velmi mladé a velmi dynamicky se rozvíjející technologické firmy, kterou však řídí velmi zkušení lidé s dlouholetou praxí a jasnou vizí. Věřím, že tato práce přispěje rukou k dílu a přinese užitek všem zainteresovaným stranám.

1 CÍLE PRÁCE, METODY A POSTUPY ZPRACOVÁNÍ

Cílem práce je návrh modulu pro propojení IS s ekonomickým softwarem včetně návrhu jeho implementace. Dílčí cíle spočívají v analýze a posouzení IS společnosti, definování požadavků a popisu principu propojení IS s ekonomickým softwarem.

Řešení má přinést automatizaci vybraného firemního procesu ve společnosti Sledovanitv.cz, s.r.o., kterým je komunikace stávajícího informačního systému s účetní agendou za účelem snížení nákladovosti a pracnosti. Řešení bude dosaženo splněním následujících úloh:

1. analýza současného stavu IS, nalezení slabin a identifikace problému, který bude předmětem řešení,
2. stanovení potřeb a požadavků společnosti a vytvoření návrhu řešení,
3. technologická implementace řešení,
4. nasazení do reálného provozu a zhodnocení nejen ekonomických přínosů.

1.1 Posouzení cílů dle SMART

Konkrétnost (specific) – cíl nyní považuji za jasně specifikovaný a bude dosažen splněním jednotlivých dílčích úloh.

Měřitelnost (measurable) – přínosy jsou reálně měřitelné, protože jsou známy hodnoty (i nefinanční), kterých je dosahováno před zásahem.

Zadatelnost (assignable) – řešení problému mi bylo přiděleno a zároveň stanoveny komunikační kanály a lidé pro realizaci konzultací. Jednoznačně jsou identifikovány také zodpovědnosti za řešení uvnitř společnosti.

Reálnost (realistic) – cílů jsem schopen dosáhnout, disponuji patřičnými zdroji, technologiemi, znalostmi i zkušenostmi potřebnými pro realizaci.

Časové ohraničení (time-bound) – společnost je závislá na uvedení řešení do provozu v zadané lhůtě a zároveň má i tato práce svůj termín vyhotovení.

1.2 Postup zpracování

V první části této práce si kladu za cíl uvést teoretické pozadí nutné pro řešení předmětné problematiky. Teoretická část se bude zakládat na třech pilířích. Záměrem první části je v logické návaznosti popsat informační systémy z analytického i implementačního pohledu a položit tak základ budoucí analýze IS a také nastínit způsoby a postupy provádění změn. Druhá část bude věnována popisu konkrétního systému (ekonomického software), kterého se bude týkat závěrečná implementace. Řeč zde kromě představení možností software bude také o technické integraci, budou představeny konkrétní principy a způsoby vzdálené komunikace a řízení. Ryze technologicky zaměřený závěr teoretické části cílí na popis programových nástrojů k vývoji a provozu samotného řešení, přičemž představí také aktuální vývojové trendy.

V analytické části bude nejprve popsána dotyčná společnost z hlediska předmětu podnikání, organizační struktury a činnosti jednotlivých štábů. Následně se pozornost obrací již výhradně na interní informační systém, který bude podroben analýze metodou HOS 8 s předpokladem, že výsledky udají směr pro závěr analytické části – analýzu změn.

Praktická část práce zaměřená na vlastní návrh řešení nejprve přinese odůvodnění výběru ekonomického software a následně se bude věnovat nejprve návrhu řešení funkcionalit a způsobu jejich implementace a dále již výhradně popisu implementace – představení jednotlivých programových komponent, metod a principů aplikační logiky a zpracování aplikace včetně ukázek důležitých či zajímavých úseků kódů demonstrujících její reálnost.

Závěrem je celý projekt podroben výstupnímu zhodnocení – představeny (nejen) ekonomické přínosy práce, uvedeny informace o reálném nasazení a provozu výstupů této diplomové práce.

2 TEORETICKÁ VÝCHODISKA PRÁCE

Cílem této části dokumentu je popis teoretického pozadí, na kterém bude postaveno řešení předmětného problému této práce. První odstavce se zabývají informačním systémem v obecné rovině, následovně pak konkrétními metodami přístupu k jeho současné analýze, výběru, nasazení a hodnocení efektivnosti. Tyto teoretické podklady se stanou zdrojem informací pro navazující kapitoly „analýza současného stavu“ a „návrh vlastního řešení“.

Záměrem druhé části teorie bude představení konkrétních technologií, na jejichž bázi bude řešení implementováno. Úmyslem zde není detailní popis všech použitých technologií, ale především výběr a uvedení stěžejních informací o vybraných softwarových prostředcích, kterých se řešení bude týkat a jejichž popis považuji pro správné informační podložení kapitoly „implementace řešení“ za nezbytný.

2.1 Informační systém

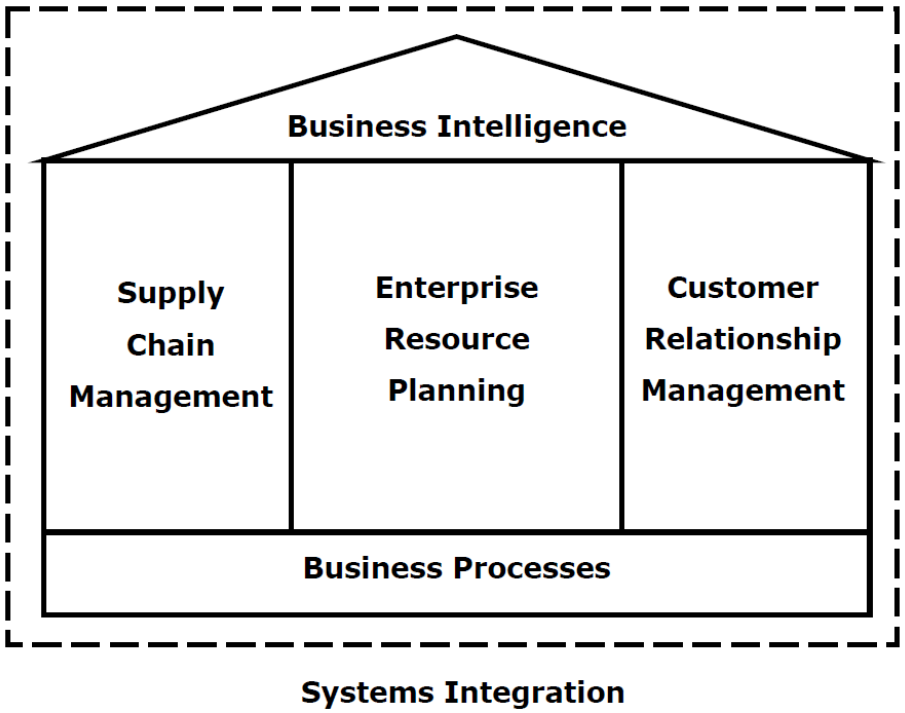
„Informační systém lze definovat jako soubor lidí, metod a technických prostředků zajišťujících sběr, přenos, uchování, zpracování a prezentaci dat s cílem tvorby a poskytování informací dle potřeb příjemců informací činných v systémech řízení.“ (Tvrdíková, 2009).

Cílem informačního systému je vždy zvýšení efektivnosti práce s informacemi. Pod spojení zvýšení efektivnosti dále spadá také termín snížení pracnosti, který byl prvním impulsem pro rozvoj informačních technologií. Informační systémy v dnešní době zajišťují řadu služeb, bez nichž si dnešní život nedovedeme představit. Předmětem této práce bude především pohled na informační systémy spadající do kategorie ERP[1].

Sodomka (2006) ERP definuje následovně: *„Informační systém kategorie ERP definujeme jako účinný nástroj, který je schopen pokrýt plánování a řízení hlavních interních podnikových procesů (zdrojů a jejich transformace na výstupy), a to ve všech úrovních, od operativní až po strategickou.“*.

Pro snadnější pochopení uvádím ERP jako součást holisticko-procesního pohledu na informační systémy (Obrázek 1).

[1] Enterprises resource planning



Obrázek 1 – Holisticko-procesní pohled IS. (Zdroj: Sodomka, 2006).

Obrázek demonstruje využívání 3 typů informačních systémů (SCM, ERP, CRM) pro zabezpečení (efektivního) řízení práce s informacemi. Sofistikovaná práce s daty (informace v elektronické podobě) nám kromě zefektivnění podnikových procesů, na kterých je postavena, přináší možnost čerpat z dat přidanou informační hodnotu prostřednictvím nástrojů business intelligence.

Customer Relationship Management (CRM) je definován jako komplex technologií (aplikační, software, technických prostředků), podnikových procesů a personálních zdrojů určených pro řízení a průběžné zajišťování **vztahů se zákazníky** podniku a to v oblastech podpory obchodních činností, zejména prodeje, marketingu, zákazníka a zákaznických služeb. *(Basl a Blažíček, 2008)*

Supply Chain Management (SCM) je definován jako **řízení dodavatelských řetězců**, eventuálně sítí, představující soubor nástrojů a procesů, které slouží k optimalizaci řízení a k maximální efektivitě provozu všech prvků celého dodavatelského řetězce s ohledem na koncového zákazníka. SCM jsou konkrétním příkladem vzájemného propojení dodavatelů s odběrateli na bázi informačních a komunikačních technologií. *(Basl a Blažíček, 2008)*

2.1.1 Životní cyklus informačního systému

Životní cyklus IS je rozdělen do 4 jednotlivých, po sobě jdoucích, na sebe navazujících a časově jednoznačně ohraničených fází, které demonstruje Diagram 1.

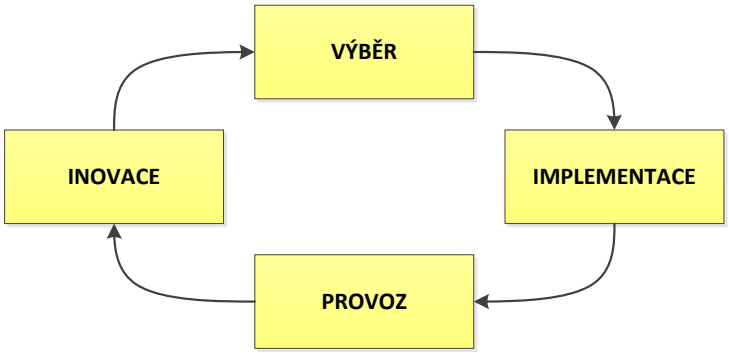


Diagram 1 – Životní cyklus informačního systému. (Zpracováno podle: Dovrtěl, 2005.)

Výběr – proces nalezení adekvátního řešení, které vyhovuje potřebám podniku z hlediska nároků na platformu, funkčnost, cenu, služby a další očekávání.
Implementace – zavedení (případně také vývoj) informačního systému, nastavení parametrů, přenos dat, nastavení podnikových procesů, stanovení práv a zodpovědností, školení uživatelů.
Provoz – testování, zajištění stabilního a kontinuálního provozu, způsob provádění údržby a řešení problémů, monitoring a vyhodnocování provozních dat.
Inovace – analyzování potřeb budoucího IS, stanovení cílů a metrik, volba dodavatele a způsobu dodání případně přechod na jiný produkt. *(Dovrtěl, 2005)*

2.1.2 Metody měření efektivnosti IS

Client impact – orientace na zákazníka je kritická pro každý podnik. Proto se i IT oddělení v podniku orientuje na projekty s velkým vlivem na zákazníka. Pomocí této techniky jsou IT projekty uspořádány do oblasti v matici například 3 x 3, kde na jedné ose je dopad (nízký, střední a vysoký) na zákazníka a na druhé ose je vliv tohoto dopadu (např. zvýšení obratu, snížení nákladů apod.). Celkový obraz pak ukazuje, zda je dostatečné množství projektů věnováno vysokému vlivu na klienta. *(Basl a Blažíček, 2008)*

Self-help – hodnota IS lze určit podle schopnosti realizovat části procesů (příp. jen jejich vybrané činnosti) automatizovaně, bez zásahu lidské obsluhy. Významný je pak podíl automatizovaných činností v poměru k ručně prováděným, což mj. umožňuje porovnat

náklady na pracovní sílu a náklady na provozování systému. Tento postup umožňuje lépe alokovat prostředky ve prospěch inovativních projektů oproti podpůrným. *(Basl a Blažíček, 2008)*

Dropsout – je technika, která měří úspěšnost nové vlastnosti webové stránky. Používají ji podniky, které velké množství interakcí uskutečňují přes web. To znamená, že poté, co byla přidána nová funkcionalita, je měřeno, kolik lidí ji úspěšně využilo, kolik bylo chybných užití a ve které části systém opustili. Tak lze odstranit problém a také rozhodnout o lepším umístění funkcionalit. *(Basl a Blažíček, 2008)*

2.1.3 Metoda HOS 8

Je metoda pro zhodnocení efektivnosti a vyváženosti podnikového informačního systému jako celku a nalezení případných nedostatků. Zpracování probíhá dotazováním kompetentních pracovníků v 8mi následujících oblastech:

1. **Hardware** – fyzické vybavení ve vztahu k spolehlivosti, bezpečnosti a použitelnosti se softwarem,
2. **Software** – programové vybavení, jeho funkce, snadnost používání a ovládání,
3. **Orgware** – pravidla pro provoz IS a doporučené pracovní postupy,
4. **Peopleware** – vliv IS na uživatele z hlediska rozvoje schopností, k jejich podpoře při užívání IS a jejich důležitosti. Nehodnotí ale odborné kvality uživatelů, ani jejich schopnosti,
5. **Dataware** – zkoumá data v IS ve vztahu k jejich správě, bezpečnosti a dostupnosti. Nehodnotí množství dat v IS, ani jejich přesnost, ale to, jak je uživatelé mohou využít a jak jsou spravována,
6. **Customers** – možnosti, které IS nabídne zákazníkovi a jak je tato oblast řízena. Nehodnotí to, jak je zákazník spokojen s IS, ale způsob řízení této oblasti IS,
7. **Suppliers** – vztah IS vůči dodavatelům, řízení dodavatelských činnosti. Nezkoumá spokojenost podniku s dodavateli, ale způsob řízení IS vzhledem k dodavatelům,
8. **Management** IS – řízení IS ve stahu k informační strategii, uplatňování pravidel a vnímání uživatelů, kteří do IS přistupují. Nezkoumá znalosti managementu IS. *(Dovrtěl, 2005).*

Každá ze zmíněných oblastí je složena z deseti otázek. Otázky jsou formulovány tak, aby byly aplikovatelné na širokou škálu informačních systémů. Snaží se cílit na konkrétní oblasti, které

bývají často problematické či kritické. Metoda využívá princip uzavřených dichotomických otázek s kvantitativní odpovědi v následujících variantách.:

ANO	SPÍŠE ANO	ČÁSTEČNĚ	SPÍŠE NE	NE

Výhodou tohoto systému dotazování pro respondenta je snadnost a rychlost odpovídání, nevyžaduje hlubší analýzu problému a formulování odpovědi.

Pro výzkumníka přináší výhodu jednoznačné interpretace odpovědi, větší ochotu respondentů, snadnost kódování a lepší možnost následného statistického zpracování.

Nominální význam hodnot u_i tj. stav zkoumané oblasti je vyjádřen hodnotou, která má následující nominální význam:

- $u_1 = 5$ znamená *velmi vysokou* úroveň oblasti *i*,
- $u_1 = 4$ znamená *vysokou* úroveň oblasti *i*,
- $u_1 = 3$ znamená *střední* úroveň oblasti *i*,
- $u_1 = 2$ znamená *nízkou* úroveň oblasti *i*,
- $u_1 = 1$ znamená *velmi nízkou* úroveň oblasti *i*.

V některých případech odpověď „ne“ znamená naopak vysokou úroveň v dané oblasti – pak je bodování dané oblasti samozřejmě inverzní. *(Dovrtěl, 2005)*

Hodnotu stavu i-té oblasti získáme po vyloučení otázky s minimálním bodovým ohodnocením a otázky s maximálním bodovým ziskem pro i-tou oblast. Po tomto vyloučení se hodnota stavu i-té oblasti vypočítá jako aritmetický průměr hodnot zbývajících otázek. Výsledná hodnota stavu i-té oblasti je získána po matematickém zaokrouhlení na celé číslo.

$$Max_i = \max\left(u_{i_1}, u_{i_2}, \dots, u_{i_{10}}\right); \quad Min_i = \mathrm{mix}(u_{i_1}, u_{i_2}, \dots, u_{i_{10}})$$

$$u_i = \left[\frac{\sum_{i=0}^{n} u_{ij} - Max_i - Min_i}{8}\right] \qquad i \in \langle 1; 8 \rangle$$

(Dovrtěl, 2005)

Souhrnný stav informačního systému lze určit pomocí vztahu s použitím hodnot z výše uvedeného podrobného stavu IS:

$$u = \min(u1, u2, \ldots, u8)$$

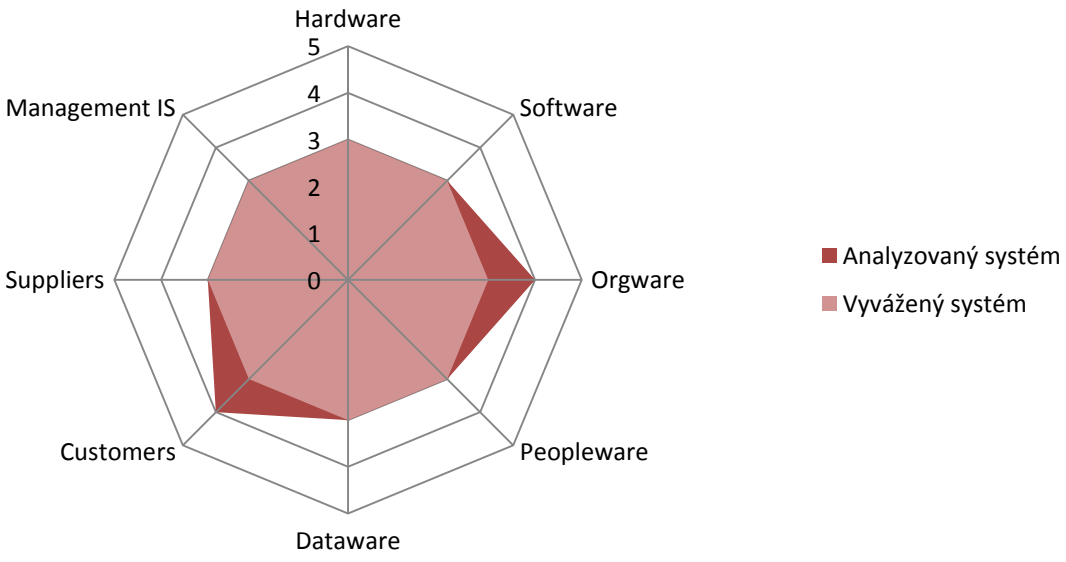
Získanou hodnotu stavu IS lze slovně interpretovat dle následující stupnice:

- $u = 5$ značí *velmi vysokou souhrnnou* úroveň stavu IS,
- $u = 4$ značí *vysokou souhrnnou* úroveň stavu IS,
- $u = 3$ značí *střední souhrnnou* úroveň stavu IS,
- $u = 2$ značí *nízkou souhrnnou* úroveň stavu IS,
- $u = 1$ značí *velmi nízkou souhrnnou* úroveň stavu IS.

  (Dovrtěl, 2005)

Za vyvážený informační systém se považuje ten, kde pro všechna u_i platí následující matematický vztah a demonstruje jej Graf 1.

$$u_i - u \leq 1 \quad \wedge \quad \sum_{i=1}^{8} (u_i - u) \leq 3$$



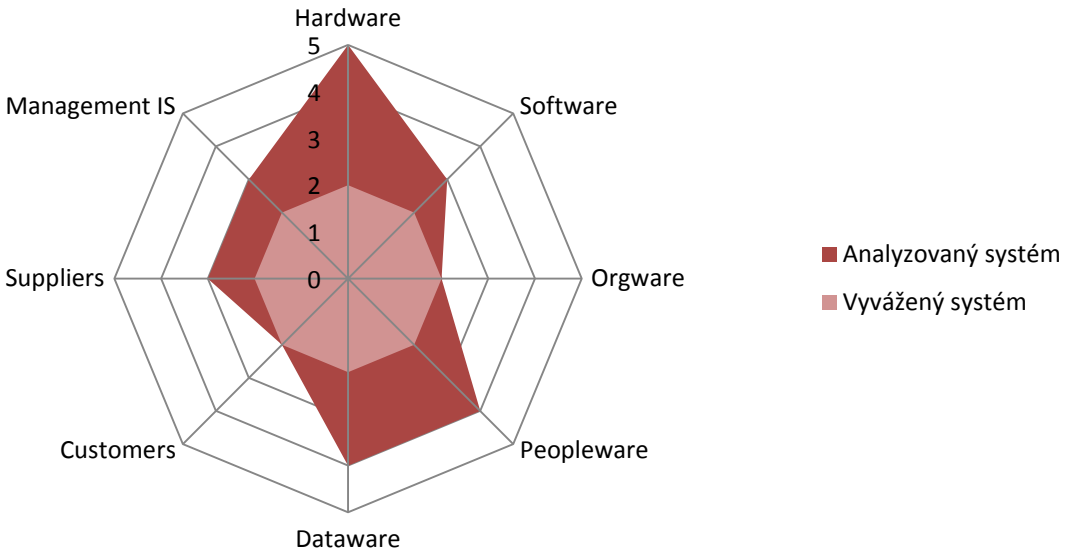
Graf 1 – Vyvážený informační systém podle HOS 8. (Zpracováno podle Koch, 2010)

Vyvážení informační systém v praxi poukazuje na jednotnou úroveň všech oblastní napříč IS. Předpokladem vyváženého systému je fakt, že žádná oblast nedosahuje nestandardně nízké

ani vysoké úrovně stavu. Uvedený zákon bude lépe zřejmý u nevyváženého informačního systému, který nyní následuje.

Za **nevyvážený informační systém** se považuje ten, pro který platí následující matematický vztah a demonstruje jej Graf 2.

$$\sum_{i=1}^{8}(u_i - u) \geq 4 \quad \vee \quad \max(u_i - u) \geq 2 \qquad 1 \leq i \leq 8$$



Graf 2 – Nevyvážený informačního systém podle HOS 8. (Zpracováno podle Koch, 2010)

Graf 2 demonstruje vysoké rozdíly mezi jednotlivými oblastmi – nejslabší oblast definuje vyvážený systém, silnější oblasti vytváření nežádoucí plochu mimo vyvážený systém.

Omezení metody HOS 8

1. Metoda neslouží k detailnímu zkoumání informačních systémů na úrovni jednotlivých procesů,
2. výsledky metody jsou založeny na subjektivních odpovědích na kontrolní otázky,
3. kontrolní otázky jsou všeobecné vzhledem k relativně širokému záběru zkoumaných informačních systémů. *(Koch, 2010)*

Tato metoda ovšem dobře poslouží pro základní analýzu a správné pochopení zkoumaného IS, nalezení slabin, které budou předmětem dalšího snažení.

2.1.4 Strategie zavádění IS

Zavedení nového IS nebo nahrazení dílčí části představuje významný zásah do již zaběhnutého procesů firmy, a proto je zvolení správné strategie kritickým faktorem úspěchu. Faktory, které výběr ovlivňují, jsou například – typ a funkce předchozího IS, objem změn a způsoby ovládání IS, připravenost jednotlivých pracovišť a pracovníků na zavedení IS atp.

Molnár (1992) rozlišuje strategie na následující:

Souběžná strategie – nový systém i ten původní jsou nějakou dobu udrženy v chodu souběžně, tím je zajištěno dostatečné otestování a proškolení uživatelů. Často ovšem toto řešení bývá technologicky náročné, případně nelze realizovat vůbec (integrita dat).

Diagram 2 – Souběžná strategie zavádění IS. (Zpracováno podle Molnár, 1992)

Pilotní strategie – informační systém je zaveden pouze pro jednu pobočku nebo uzavřenému okruhu uživatelů, na kterých se funkce otestují. Po vyladění je pak systém nasazen plošně.

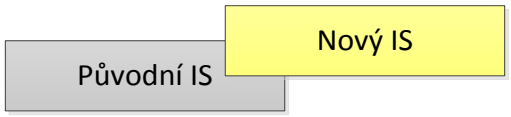


Diagram 3 – Pilotní strategie zavádění IS. (Zpracováno podle Molnár, 1992)

Postupná strategie – principem je postupné nahrazování jednotlivých modulů. Každý modul je testován zvlášť a úspěšným provozem jednoho modulu se postupuje k modulu následujícímu. Tento postup je výhodný především u rozsáhlých systémů, kde existuje vysoká úroveň modularity.

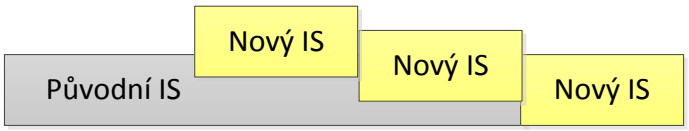


Diagram 4 – Postupná strategie zavádění IS. (Zpracováno podle Molnár, 1992)

Nárazová strategie – vypnutí původního systému a spuštění nového systému. Odladění je pak prováděno za ostrého provozu. Tato rizikovou strategii je možno provádět u systémů, nebo částí, které nejsou pro chod organizace kritické.

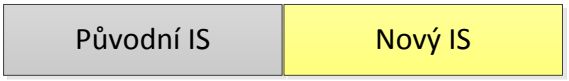


Diagram 5 – Nárazová strategie zavádění IS. (Zpracováno podle Molnár, 1992)

V praxi většinou nastává nutnost uvedené postupy kombinovat. Nejčastější je kombinace postupu nárazového a postupného. V implementační části této práce se předpokládá právě postupné otestování jednotlivých částí na reálných datech a jednorázové spuštění vyladěné aplikace.

2.2 Ekonomický systém FlexiBee

FlexiBee je moderní ekonomický systém určený pro malé a střední firmy či firmy vedoucí účetnictví a pro drobné podnikatele vedoucí daňovou evidenci. Jedná se o plně lokalizovatelný produkt s podporou vzdáleného přístupu. Funkčností pokrývá podvojné účetnictví, homebanking, skladové hospodářství, evidenci majetku a leasingu, mzdy, pohledávky/závazky a má řadu dalších modulů. Program poskytuje intuitivní uživatelské rozhraní s podporou evropské legislativy. FlexiBee je dostupný pro operační systémy Windows, Linux a Mac OS X. *(FlexiBee Systems, 2014)*

2.2.1 Vlastnosti ES

Multiplatformita - FlexiBee podporuje Windows, Linux a Mac OS X bez rozdílu na serverech i na klientech.

Cloudové uložiště - FlexiBee online podporuje možnost provozu v cloudu – zajišťuje tak kompletní provoz služby. Ke svým datům mohou klienti přistupovat odkudkoliv z počítače či notebooku, či prostřednictvím aplikací pro smartphony na platformách iOS a Android. Data jsou bezpečně uložena na serveru a tak odpadá nebezpečí krádeže zařízení. Mimo to je systém možné provozovat i na vlastních serverech.

Bezpečnost

a. **Šifrování** – FlexiBee používá silné šifrování a postupy založené na SSL[2] protokolu.
b. **Sledování změn v systému** – všechny změny v systému se zapisují do záznamu o změnách (log). Tento záznam nelze uživatelsky měnit ani mazat.
c. **Víceúrovňové zamykání dokladů a období** – protože se systémem může pracovat mnoho uživatelů současně, je potřeba zajistit, aby dokladu zpracované účetní již nikdo neměnil. Proto lze zamknout jednotlivé doklady, ale i celá období (např. kvůli DPH).
d. **Automatické zálohování** – na serverech jsou prováděny inkrementální zálohy několikrát denně bez ohledu na to, zda klient využívá službu v cloudu nebo provoz zajišťuje vlastní režií serverů.
e. **Zabezpečení serverů** – fyzický přístup k serverům je omezen jen na povolené pracovníky a přístup je monitorován 24 hodin denně s fyzickou ostrahou na místě. Žádný pracovník nemůže přistupovat přímo ke klientským datům. Pokud technické podpoře klient neposkytne zálohu sám, nebude k ní mít ona přístup. Na serverech, kde běží služby FlexiBee online neběží žádné jiné služby. Zálohy jsou umístěny na záložních systémech s vysokým zabezpečením.

Rozšiřitelnost

f. **Doplňující moduly** – FlexiBee obsahuje mnoho modulů k dokoupení – od maloobchodní kasy až po napojení na systémy dodavatelů.
g. **Analytické nástroje** – systém používá uložiště na bázi SQL[3]. Z datových skladů je možno dolovat data a aplikovat metody Business Inteligence[4].
h. **Propojené systémy –** Flexibee již má integraci na různé elektronické obchody, CRM[5], dotykové pokladny, registrační pokladny: pokladní systémy AWIS, skladový systém SOFIX, SCUDe CRM, TeamOnline CRM.

(FlexiBee Systems, 2014)

[2] Secure Sockets Layer – protokol pro zabezpečení komunikace šifrováním a autentizaci komunikujících stran.
[3] Structured Query Language - dotazovací jazyk pro práci s daty v relačních databázích.
[4] Business Inteligence - dovednosti, znalosti, technologie, aplikace, kvalita, rizika, bezpečnostní otázky a postupy používané v podnikání pro získání lepšího pochopení chování na trhu a obchodních souvislostí.
[5] Customer relationship management – managementu řízení vztahů se zákazníky.

2.2.2 Funkce systému

Řízení obchodních partnerů – klienti mohou evidovat informace o firmě, místech určení a kontaktech či přikládat přílohy s dokumenty (např. smlouvy). Dále také vést jednotlivé pobočky zákazníků (místa určení) a na všech datech provádět rozmanité analýzy. Flexibee podporuje systém hodnocení kreditibility partnerů CreditCheck[6].

Skladová evidence – software nabízí možnost vedení neomezeného množství skladů. A díky online cloudovému uložišti má klient vždy aktuální přehled o všech svých pobočkách. A to včetně výrobních čísel, expirací (trvanlivosti) či šarží.

Účetnictví – zajišťuje kompletní účtování a to nejen v rámci ČR ale také EU. Automaticky řeší řadu rutinních úkonů (např. kurzové rozdíly). Systém nabízí možnost zamykání dokladů a období. Po uzamčení již nikdo nemůže provádět změny a tím ovlivnit výsledek bez vědomí účetní. Pro všechny úkony nabízí důkladné členění dle mnoha kritérií (činnosti, střediska či zakázky atp.). FlexiBee umí pracovat všechny potřebné výkazy a přiznání: rozvaha, výsledovka, peněžní deník, DPH, souhrnné hlášení, intrastat, ISPV, IFRS).

Personalistika a mzdy – zahrnuje vypočet mezd včetně všech změn, odměn, srážek, přirážek, odrážek a k tomu si vést personální informace. Dále umí správně započítávat mzdové náklady na zakázky či střediska. Mzdy vypočítává také do budoucna, což klientům dává lepší představu o mzdovém cashflow včetně předpokládaných výpovědích či prémiích. Pokud klient kombinuje více pracovních poměrů u jednoho zaměstnance, FlexiBee umožní jejich výpočet. Klient tak například může kombinovat dohodu o provedení práce s klasickou mzdou. *(FlexiBee Systems, 2014)*

2.3 REST architektura webového API

REST (representational state transfer) – je architektura rozhraní, navržená pro distribuované prostředí. Pojem *REST* byl představen v roce 2000 v disertační práci Roye Fieldinga, jednoho ze spoluautorů protokolu *HTTP*. *REST* je, na rozdíl od známějších *XML-RPC* či *SOAP*, orientován datově, nikoli procedurálně. Webové služby definují vzdálené procedury a protokol pro jejich volání, *REST* určuje, jak se přistupuje k datům. *REST* tedy poskytuje prostředí pro komunikaci a výměnu dat mezi aplikacemi prostřednictvím *HTTP* protokolu.

[6] Credity check - společnost poskytující služby ověřování kreditibility obchodních partnerů.

REST svou bezstavovostí vychází vstříc moderním metodám vývoje webových aplikací, které jsou založené na paralelním zpracování distribuovaného obsahu. *(Malý, 2009)*

2.3.1 Metody přístupu

REST implementuje čtyři základní metody přístupu (nebo také dotazovací metody), které jsou známé pod označením *CRUD*:

1. ***CREATE*** – metoda požadavku na vytvoření dat,
2. ***RETRIEVE*** – metoda požadavku na získání dat *(get)*,
3. ***UPDATE*** – metoda požadavku pro úpravu dat (*put*),
4. ***DELETE*** – metoda požadavku na odstranění odstranění dat.

GET (retrieve) - Jedná se o základní metodu, která zapouzdřuje požadavky na získání dat. Typickým příkladem využití metody *GET* je zaslání http požadavku na získání webové stránky webovému serveru:

```
GET /zbozi/boty.html
Host: api.boty.cz
```

Každý zdroj (recource) má podle *REST* rozhraní vlastní identifikátor (*URI*). Prostřednictvím *HTTP GET* požadavku pak dochází k získání konkrétního zdroje. Konkrétní příklad získání záznamů z Flexibee adresáře ve formátu *JSON* může vypadat následovně:

```
GET /c/uri_identifikator_spolecnosti/adresar/
Host: 81.201.202.203:5434
HEAD Accept: application/json
```

Úpravou *GET* požadavku (doplněním o identifikátor) můžeme získat konkrétní záznam:

```
GET /c/uri_identifikator_spolecnosti/adresar/54255.json
```

POST (create) - Metoda POST je obecně metodou, kterou obecně využívají například webové formuláře pro odeslání uživatelských vstupů. Ve chvíli volání nelze využít konkrétní identifikátor (viz metoda GET) protože tento bude záznamu přidělen vytvořením.

Po odeslání server vrací patřičný **návratový kód** – *HTTP* má k tomu účelu stavový kód (2xx – success) 201 – *Created*, v němž lze předat *URI* nově vytvořeného zdroje. Pokud došlo k chybě, měl by server vrátit chybový kód (5xx – server errors).

DELETE - Zdroj lze smazat pomocí volání *URI HTTP* metodou *DELETE*. Volání je obdobné volání metody *GET*:

```
DELETE /c/uri_identifikator_spolecnosti/adresar/5425
Host: 81.201.202.203:5434
```

V praxi bývá někdy problematické vyvolat *HTTP* metodu *DELETE* – spousta *HTTP* nástrojů či *HTML* formuláře jsou omezeny pouze na metody *POST* a *GET*. V praxi se proto u *REST* rozhraní používají náhradní způsoby – např. volání pomocí *POST* s parametrem, který sděluje, že má být ve skutečnosti použita metoda *DELETE*, nebo speciální *URI*.

PUT (update) - Operace změny je podobná operaci vytvoření (*CREATE*, metoda *POST*), s tím rozdílem, že voláme konkrétní *URI* konkrétního zdroje, který chceme změnit, a v těle předáme novou hodnotu (jako u metody *POST*). Na rozdíl od *POST* je u úprav zdroje jeho *URI* už známá, takže ji lze zadat. REST-API a FlexiBee. *(Malý, 2009*)

Struktura URL pro FlexiBee se sestavuje z několika částí:

```
/c/<identifikátor firmy>/<evidence>/<ID záznamu>.<výstupní formát>
```

Identifikátor firmy - při komunikaci se serverem je nutné uvádět identifikátor firmy. Ten obvykle vychází z názvu firmy, ale není to podmínkou.

Evidence - Volí se typ evidence (prostřednictvím klíčového slova), se kterou má systém pracovat. Přehled všech evidencí FlexiBee je uveden v příloze.

ID Záznamu - záznamy ve FlexiBee je možné identifikovat několika způsoby

Název	Popis	Ukázka
FlexiBee ID	Jednoznačný a neměnný FlexiBee identifikátor.	`123`
Kód/Značka	Uživatelské označení. Přiděleno uživatelem s možností změn.	`code:CZK`
Key	Unikátní náhodný identifikátor (atribut UUID). Nelze jej měnit.	`key:550e8400`
PLU	PLU je identifikační kód, který se používá při prodeji. Obvykle se jedná o 4- či 5místný číselný kód.	`plu:4020`
EAN	Záznam je identifikován podle kódu EAN (čárový kód).	`ean:4710937332698`
Externí identifikátor	Identifikátor z externí aplikace. V aplikaci jej nelze měnit. Lze měnit z externích systémů.	`ext:SHOP:123`
VAT ID	Identifikátor dle daňového čísla. V ČR odpovídá DIČ, v SR odpovídá IČ DPH.	`vatid:CZ28019920`
IČ	Identifikátor dle IČ.	`in:28019920`

Tabulka 1 – Způsoby identifikace záznamů ve Flexibee. (Zdroj: flexibee.eu/api/doc/ref/identifiers)

Výstupní formát

REST API FlexiBee dokáže exportovat a importovat data v různých formátech. Typ výstupního formátu může být určen příponou (.xml, .json) nebo pomocí hlavičky v HTTP požadavku (Accept). Formát jako přípona je nepovinný a má přednost před HTTP hlavičkou. Pokud je uvedeno Accept: */* je vrácena HTML forma.

FlexiBee podporuje následující formáty:

Název	Poznámka	Content-Type	Import
HTML	HTML stránka pro zobrazení informací na webové stránce.	`text/html`	ne
XML	Strojově čitelná struktura ve formátu XML.	`application/xml`	ano
JSON	Strojově čitelná struktura ve formátu JSON.	`text/javascript, application/json`	ano
CSV	Tabulkový výstup do formátu CSV (Column Separated Values).	`text/csv`	ano
DBF	Databázový výstup ve formátu DBF (dBase).	`application/dbf`	ano
XLS	Tabulkový výstup ve formátu Excel.	`application/ms-excel`	ano
ISDOC	e-faktura ISDOC.	`application/x-isdoc(x)`	zatím ne (jen z aplikace)
EDI	Elektronická výměna data (EDI) ve formátu INHOUSE.	`application/x-edi-inhouse`	jen objednávky přijaté
PDF	Generování tiskového reportu.	`application/pdf`	ne
vCard	Výstup adresáře do formátu elektronické vizitky vCard.	`text/vcard`	ne
iCalendar	Výstup do kalendáře ve formátu iCalendar.	`text/calendar`	ne

Tabulka 2 – Podporované vstupně/výstupní formát. (Zdroj: flexibee.eu/api/doc/ref/format-types)

Výpis evidence - neuvedením identifikátoru záznamu vrátí server všechny položky evidence (jejich výpis). Příklad použití:

```
/c/<identifikátor firmy>/<evidence>
```

Použití **filtru** (specifikace podmínek výsledků):

```
/c/<identifikátor firmy>/<evidence>/(<filtr>)
```

Sumarizace záznamů - pro získání základní sumace o dané evidenci, použijte sumaci:

```
/c/<identifikátor firmy>/<evidence>/$sum
```

Filtry a sumarizaci je možné **kombinovat** sestavením *URI* v následujícím formátu:

```
/c/<identifikátor firmy>/<evidence>/(<filtr>)/$sum
```

Přehled atributů - pro každou evidenci je možné získat seznam atributů, které tato evidence podporuje. Tento přehled zohledňuje přístupová práva a licencování.

```
/c/<identifikátor firmy>/<evidence>/properties
```

Přehled tiskových reportů - u evidence je možno získat seznam podporovaných reportů pro tisk do PDF:

```
/c/<identifikátor firmy>/evidence>/reports
```

Přehled pod-evidencí - každá evidence může mít pod-evidenci (relaci). Příkladem může být položka faktury nebo kontakty u adresáře. Tyto záznamy jsou obvykle přístupné i přímo jako evidence. Rozdíl je v tom, že podevidence je filtrována danou relací. Přehled podevidencí lze získat takto:

```
/c/<identifikátor firmy>/<evidence>/relations
```

S pod-evidencemi lze pak pracovat stejně jako s evidencí:

```
/c/<identifikátor firmy>/<evidence>/<ID záznamu>/<podevidence>
```

Při exportu z FlexiBee lze spolu s konkrétním záznamen exportovat i jeho pod-evidence, použitím parametrů:

```
?relations=vazby,prilohy,bankovniSpojeni.
```

Sestavování *HTTP* požadavků bude kritickou funkcionalitou vyvíjené aplikace v rámci implementace řešení a uvedené informace poslouží jako rychlá integrační příručka.

Poznámka: uvedené ukázky čerpají z programátorské API dokumentace společnosti Flexibee Systems a jsou online dostupné na adrese: flexibee.eu/api/doc/ref/urls.

2.4 Nette framework

Nette je název frameworku, softwarové struktury, která slouží jako podpora při organizaci, vývoji a programování aplikací běžících na bázi skriptovacího programovacího jazyka *PHP*, který je určen především pro vývoj dynamických webových aplikací. Interpret jazyka *PHP* je nejčastěji využíván v kombinaci s webovým serverem *Apache* (případně *Nginx*, *LightHTTP*, a další). Pro uchovávání dat framework využívá vlastní databázovou vrstvu *NDB* (implementuje knihovnu *NotORM*) a drivery pro připojení rozmanitého množství databází, přičemž mezi vývojáři bezkonkurenčně vede databázový systém *MySQL* (případně jeho fork *MariaDB*). Uvedená kombinace technologií nese označení *LAMP* (*Linux, Apache, MySQL, PHP*) a je dnešním (2014) lídrem na poli webových server-side technologií. *(Nette Foundation, 2014)*

Nette je moderní framework s promyšleným a čistým objektovým návrhem, využívajících moderních vlastností *PHP 5* (v dnešní době se bavíme o verzích 5.3 a 5.4), komponent a událostmi řízeného modelování. Podporuje moderní koncepty, praktiky vývoje webových aplikací postavených na technologiích *AJAX/AJAJ*[7], *Dependency Injection*[8], *SEO*[9], *DRY*[10], *MVC*[11], *Web 2.0*[12]. *(Nette Foundation, 2014)*

2.4.1 Bezpečnost

Mimo moderní technologie Nette eliminuje vznik a výskyt bezpečnostních děr a jejich zneužití. Bezpečnost je dnes velmi důležitým tématem a proto zde uvádím alespoň základní informace o zabezpečení frameworku.

Cross-Site Scripting (*XSS*) - je metoda narušení webových stránek zneužívající neošetřených výstupů. Útočník pak dokáže do stránky podstrčit svůj vlastní kód a tím může stránku pozměnit nebo dokonce získat citlivé údaje o uživatelích. Proti XSS se lze bránit jen důsledným a korektním ošetřením všech vstupních řetězců. Příkladem útoku může být podstrčení upraveného hodnoty parametru *URL* adresy obsahující například JavaScriptový

[7] Asynchronous JavaScript and XML / JSON – asynchronní komunikace prostřednictvím JavaScriptu.
[8] Dependency injection - objektově orientovaná programovací technika pro vytváření závislostí (využívání) jednotlivých komponent programu bez předchozí reference.
[9] Search Engine Optimalization – metodika pro usnadnění indexování webové stránky vyhledávacím enginem.
[10] Don't Repeat Yourself – princip vývoje aplikace bez opakování již jednou napsaného zdrojového kódu.
[11] Model-view-controller – oddělení datového modelu aplikace od uživatelského rozhraní a řídící logiky.
[12] Etapa vývoje webu, kdy je statický obsah nahrazen obsahem dynamickým, který tvoří samotní uživatelé.

kód. Pokud tuto hodnotu bez ošetření interpret vloží do *HTML* výstupu, je kód v prohlížeči vykonán a útok se stává zdařeným.

Nette framework implemetuje technologii *Context-Aware Escaping*, která výstupy ošetřuje automaticky (také v závislosti na části dokumentu, ve které je výstup použit), tím odpadá práce a riziko na ručním ošetřování výstupů a vzniku bezpečnostních mezer.

Cross-Site Request Forgery (*CSRF*) - někdy označováno také jako *XSFR*, je typ útoku vedený proti aplikacím, které pro uchovávání přihlašovacích údajů využívají soubory cookie. Útočník, který zná strukturu aplikace, může uživatele (případně administrátora) donutit (přidáním parametrů *GET* do *URL* adresy) přistoupit na zamýšlenou stránku, nejčastěji její administrační časti a nevědomky provést akci (uživatel je již dříve přihlášen, server požadavek autorizuje a akci vykoná). Tento typ útoků se tedy týká aplikací s nechráněným *URL*. Proti útoku se lze bránit generováním a ověřováním autorizačního tokenu, který v případě Nette spočívá v jednoduchém volání metody *addProtection()* nad generovaným formulářem. *(Nette Foundation, 2014)*

2.4.2 Licenční politika

Nette Framework je šířen jako svobodný software. Uživatelé mají sami na výběr z licencí New BSD nebo GNU General Public License (GPL) ve verzi 2 nebo 3. Licence BSD je doporučena pro většinu projektů, jelikož je snadné ji pochopit a neklade téměř žádná omezení na to, co můžete s frameworkem dělat. Je povoleno používáné též v komerčních projektech. Název frameworku je chráněn ochrannou známkou.

2.4.3 Architektura MVC

Architektura *MVC* (Model-view-controller) v posledních letech dramaticky nabírá na popularitě, což je pravděpodobně způsobeno dosaženého stavu, kdy jednotlivé technologie již mají ranou fázi vývoje za sebou a vedle „základní výbavy“ se tak snaží nabídnout i prostředky pro tvorbu aplikací s kvalitní architekturou.

Architektura MVC dělí aplikaci na 3 logické části tak, aby je šlo upravovat samostatně a dopad změn byl na ostatní části co nejmenší. Tyto tři části jsou *model*, *view* a *controller*. *Model* reprezentuje data a business logiku aplikace, *view* zobrazuje uživatelské rozhraní a *controller* má na starosti tok událostí v aplikaci a obecně aplikační logiku.

Model – ve webovém prostředí je identický s modelem v desktopových technologiích. (Obsahuje data a business logiku, s konkrétní prezentací nemá nic společného.)

View – je v prostředí webu serverová aplikace, která zajišťuje generování *HTML* výstupu. Generátorem pak na serverové straně nemusí být nutně *PHP* ale například také *C#, Java, Ruby* a podobně. Navíc ani u webové aplikace není nutné, aby bylo výstupem vždy *HTML* – klidně to mohou být formáty jako *XML* nebo *JSON*.

Controller – se v prostředí webu nejčastěji skládá ze dvou hlavních částí. První je tzv. *Front Controller*, který zachytává všechny *HTTP* požadavky, ty následně zpracuje a přepošle konkrétním *controllerům*, což je ona druhá část. Konkrétní *controller* potom typicky přijme data původně pocházející z *HTTP* požadavku, uloží je do modelu a ten prováže s konkrétním *view*, které už se umí o vyrenderování obsahu do *HTML* či jiného formátu postarat. *(Borek, 2009)*

2.4.4 Architektura MVP

Nette framework využívá *MVP* architekturu. *MVP* vychází z *MVC*, některé myšlenky rozšiřuje, doplňuje či využívá odlišnými způsoby. Rozdíly jsou následující:

1. Uživatelský vstup i výstup plně kontroluje *view* (skrze ovládací prvky uživatelského rozhraní jako button nebo text-input).
2. Primární motivací pro oddělení *view* a *presenteru* už není nutnost ošetření vstupu, důvody jsou čistě architektonické (prezentační vrstva s *MVP* se udržuje lépe než monolitické *view*).
3. *View* má typicky přímou vazbu na *presenter*. Většinou to není technicky nutné, ale řada lidí tuto realizaci preferuje a i většina literatury o *MVP* s přímou vazbou *view* na *presenter* počítá.
4. *Presenter* v mnoha případech přímo pracuje s *view*, takže i tato vazba je silnější než v případě vzoru *MVC*.

Model – zcela vychází z *MVC* = data plus business logika.

View - navíc oproti *MVC* zpracovává uživatelský vstup. Typicky dělá pouze to, že např. v reakci na kliknutí myši zavolá nějakou metodu na Presenteru – pouze deleguje uživatelské akce. Jakákoliv aplikační logika ve *view* je chybou.

Presenter - obsahuje aplikační a prezentační logiku. Manipuluje s modelem, což pomocí systému notifikací zajistí aktualizaci *view*, nebo ovlivňuje *view* přímo. *(Borek, 2009)*

Následující Diagram 6 demonstruje rozdíly mezi oběma popsanými architekturami.

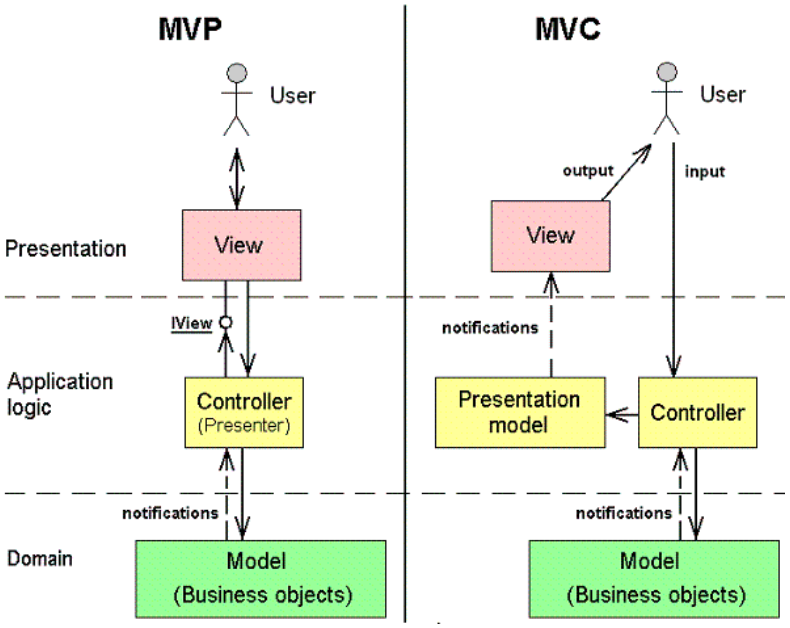


Diagram 6 – Porovnání architektur MVC a MVP. (Zdroj: c-sharpcorner.com)

Komponenty

Komponenta je zjednodušeně část webu, která obstarává právě jednu komplexní funkci a zároveň část, pro kterou může být výhodná znovu-použitelnost.

Celý proces použití komponenty se pokusím přiblížit na příkladu sestavení přihlašovacího formuláře (včetně funkce autorizace uživatele). Uživatel voláním stránky, která má formulář obsahovat, vytvoří požadavek. Tohoto požadavku se ujme *presenter*, který zjistí, že uvedená stránka obsahuje komponentu přihlašovacího formuláře a zadává *komponentě* požadavek na jeho obsluhu. *Komponenta* požadavek přijme (včetně vstupních dat) a ve spolupráci s připojenými *modely* celou událost obslouží. Touto obsluhou může být například přijetí uživatelských dat a jejich odeslání do modelu, který provádí autorizaci. *Model* porovná přijatá data s těmi, které jsou uloženy v databázi a v případě úspěchu uživatele přihlásí. Informaci o výstupu své práce předává zpět komponentě, která jej zpracuje a svůj výstup odešle do své interní šablony. Celý tento výsledek pak *presenter*, včetně dalšího požadovaného obsahu vloží do hlavní šablony (data jsou naroubována do *latte* šablony) a kompletní stránka je odeslána zpět klientovi.

3 ANALÝZA PROBLÉMU

Úlohou této kapitoly je v první řadě analýza a popis prostředí, pro které bude řešení vytvářeno. Zde se jedná nejprve o popis firmy jako takové, její předmět podnikání, organizační členění, rozdělení a povinnosti jednotlivých štábů. V návaznosti pak bude provedena analýza současného stavu informačního systému metodou HOS 8 a její vyhodnocení. Závěr kapitoly se zaměří na analyzování změn, které mají být uskutečněny.

3.1 O společnosti

Sledovanitv.cz je relativně mladý technologický projekt. Společnost Sledovanitv.cz s.r.o., která celý projekt zastřešuje, oficiálně funguje teprve první rok. Celý koncept však existuje již řadu let, a několik let je vyvíjen také informační systém, který je základním pilířem celého projektu. V práci budu dále používat pouze zkrácený název – Sledovanitv.cz.

Sledovanitv.cz se zabývá poskytování televizního vysílání svým partnerům (B2B), kteří toto dále distribuují jako službu svým zákazníkům, kde slovem zákazník je uvažován koncový konzument televizního obsahu (divák). Typickými partnery jsou firmy, které televizní vysílání poptávají jako (často) dodatkovou službu pro své zákazníky. Nejčastějším hlavním předmětem podnikání partnerů je poskytování internetové konektivity. Televizní vysílání pak pro koncového zákazníka tvoří součást dodávky širokopásmového připojení.

Projekt Sledovanitv.cz byl založen třemi společníky, kteří již řadu let spolupracují v rámci jiné společnosti – SychrovNET s.r.o., která je již více než deset let zavedeným poskytovatelem internetového připojení pro zákazníky v lokalitě města Vsetín a Zlínského kraje. Letité zkušenosti jsou nyní promítnuty ve zcela novém a ojedinělého konceptu poskytování televizního obsahu.

Společnost Sledovanitv.cz své služby provozuje v rámci Jihomoravského inovačního centra v prostorách chráněného prostředí Technologického inkubátoru Vysokého učení technického v Brně, který je určen pro nově vznikající projekty s inovačním potenciálem[13].

[13] Jihomoravské inovační centrum, zdroj: jic.cz

3.1.1 Předmět podnikání

Společnost podniká v oboru služeb spojených s informačními technologiemi. Předmětem podnikání je primární distribuce televizního vysílání – *IPTV* (šíření digitální televize prostřednictvím *IP protokolu*). Tuto službu společnost provozuje prostřednictvím systému, který sama vyvíjí a spravuje tak, aby minimalizovala nároky na hardwarové a softwarové zásahy do síti a systémů partnerů. Technická spolupráce mezi Sledovanitv.cz a jejími partnery probíhá přes *API* a webové rozhraní.

3.1.2 Personál a organizační struktura

Společnost lze na základě hierarchie pracovníků označit za firmu s liniově štábní organizační strukturou s vysokou úrovní provázanosti. To v praxi znamená, že jeden pracovník často v rámci svojí práce obstarává více činností, které spadají do různých logických oddělení společnosti (štábů). Pro identifikaci klíčových podnikových procesů je vhodné zakreslit organizační rozložení do následujícího diagramu (Diagram 7).

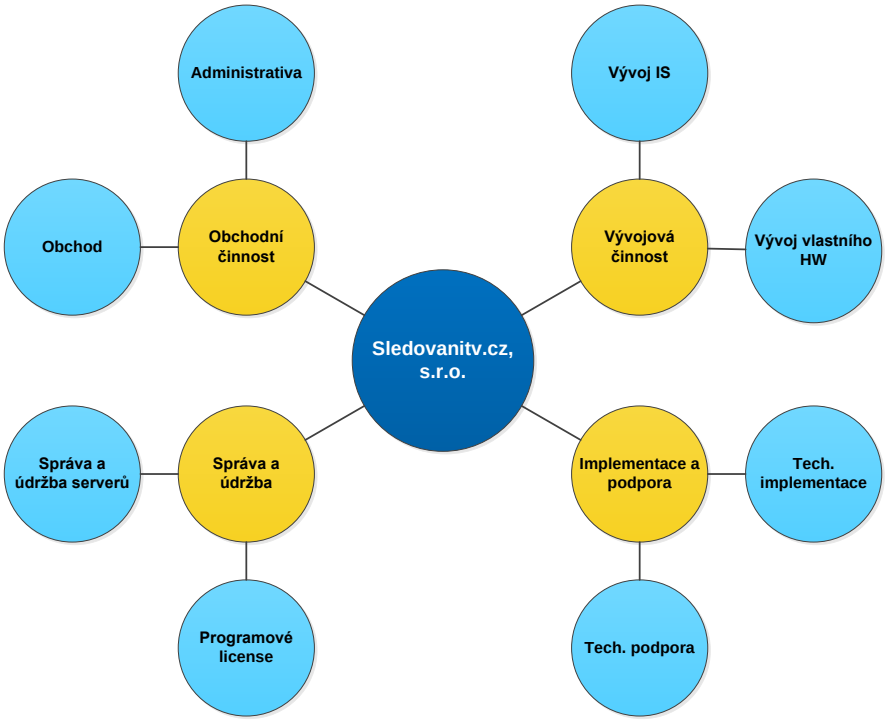


Diagram 7 – Logické členění podniku.

V další části práce popisuji jednotlivé štáby společnosti a provázání pracovníků mezi nimi. Toto považuji za důležité z hlediska správného určení zainteresovaných osob pro budoucí rozvoj informačního systému.

3.1.3 Vývojová činnost

Vývojová činnost společnosti se ubírá dvěma proudy. Primárně jde o vývoj vlastního informačního systému, který zabezpečuje poskytování služeb společnosti. Sekundárně pak o vývoj vlastního hardware, konkrétně set-top boxů.

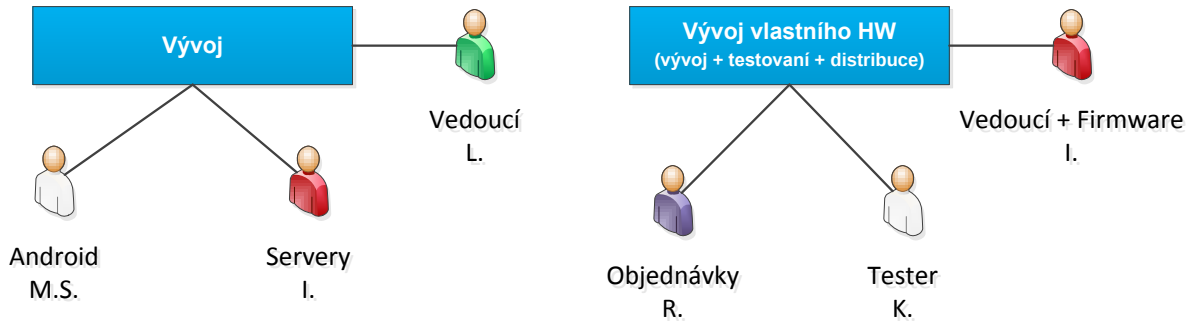


Diagram 8 – Personální zajištění vývojové činnosti.

Vývoj vlastního informačního systému je pak organizován dle zařízení a platformy. Tento vývoj zabezpečují 3 pracovníci, přičemž vedení a vývoj systému na bázi webového rozhraní zajišťuje jeden pracovník, backendovou část serverových zařízení druhý pracovník a třetí vývoj systémů pro mobilní zařízení. Zde nutno podotknout, že společnost část svého vývoje outsourcuje (například vývoj pro další platformy mobilních zařízení).

Vývojem vlastního hardware je uvažován vývoj a distribuce vlastních set-top boxů (které jsou obvykle nabízeny koncovým zákazníkům jako maximálně kompatibilní řešení). Toto oddělení musí zajistit více procesů:

1. objednávek fyzických komponent,
2. vývoj firmware,
3. testování.

3.1.4 Obchodní činnost

Obchodní oddělení zabezpečuje vyjednávání a komunikaci s partnery. Partnery jsou primárně poskytovatelé obsahu koncovým zákazníkům. Obchodní činnost ve společnosti zajišťují dva pracovníci, přičemž je nutná velmi blízká spolupráce s vývojovým oddělením, které služby následně zajišťuje. Konzultantem pro obchodní oddělení je vedoucí pracovník vývojového oddělení.

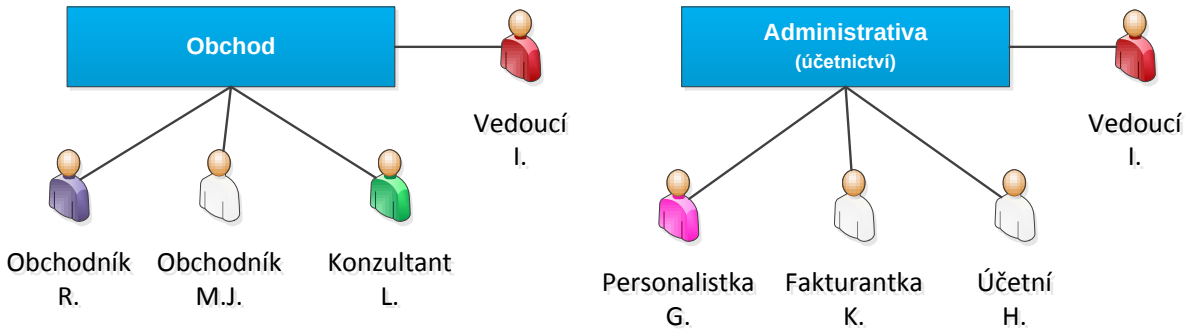


Diagram 9 – Personální obsazení obchodní činnosti.

Na obchodní činnost navazuje administrativní oddělení. Administrativní činnosti trvale zabezpečují dva pracovníci – personalistka a fakturantka, externě pak spolupracuje účetní. Tato diplomová práce se bude dále zabývat především analýzami a návrhy na úrovni administrativní a vývojové činnosti.

3.1.5 Technická implementace a podpora

Smlouvy uzavřené obchodním oddělením dále postupují technici, kteří požadovanou službu partnerům instalují, nastaví a zprovozní.

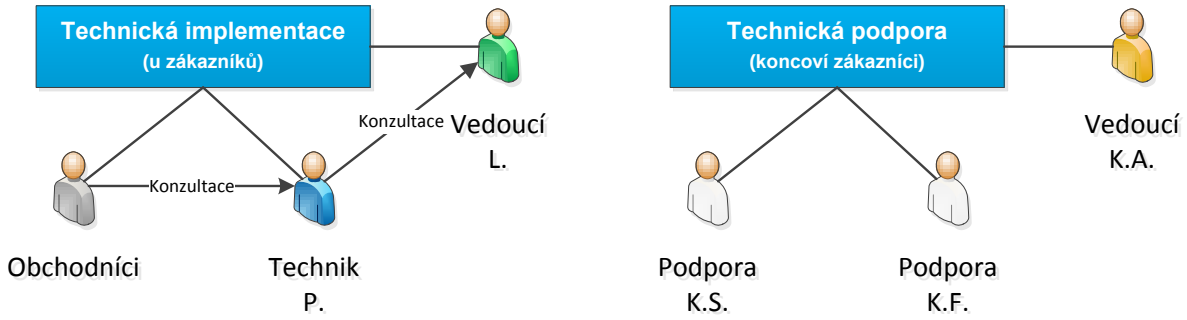


Diagram 10 – Personální obsazení implementačních činností a podpory.

Tento proces je neustále periodicky se opakující činností, kterou mohou zajistit a zajišťují sami obchodníci. Ti v případě problémů komunikují s technikem. Pokud technik narazí na vážnější problém, tak se odvolává výše na vedoucího pracovníka vývoje. Společnost poskytuje vlastní technickou podporu koncovým zákazníkům. Tuto činnosti zajišťují 3 pracovníci.

3.1.6 Správa a údržba

Velmi důležitou činností pro fungování společnosti je zajištění nepřetržitého provozu serverů, na kterých běží její služby. Správu a údržbu zařízení zajišťují dva technici. Kromě samotných serverů zde patří též správa, instalace a aktualizace všech síťových zařízení.

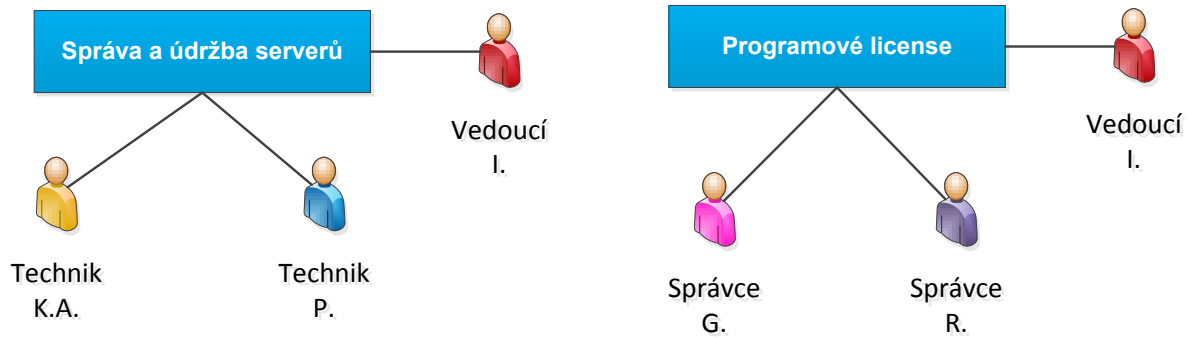


Diagram 11 – Personální zajištění správy a údržby.

Hlavním předmětem podnikání společnosti je poskytování televizního obsahu, který je sám o sobě velmi variabilně a přísně licencován. O kontinuální zajištění všech příslušných povolení a opatření zajišťují dva pracovníci.

3.2 Hodnocení IS metodou HOS 8

Pro analýzu současného stavu informačního systému společnosti Sledovanitv.cz jsem zvolil metodu HOS 8. Metoda byla volena především z důvodu komplexního uchopení informačního systému jako celku a pohled na něj z hlediska jeho různých oblastí:

1. Hardware – fyzické komponenty, zajišťující chod IS,
2. Software – programové vybavení a úroveň IS,
3. Orgware – práva a zodpovědnosti pracovníků vůči IS,
4. Peopleware – uživatelé IS,
5. Dataware – data IS,
6. Customers – interakce IS se zákazníky,
7. Suppliers – interakce IS s dodavateli,
8. Management IS.

V další části této práce jsou uvedeny výsledky analytického šetření, které jsou závěrem vyhodnoceny.

3.2.1 Hardware

První zkoumanou oblastí je hardware. Společnost se primárně snaží zaměřovat na high-end technologie a v této oblasti vykazuje nadprůměrné výsledky. Servery i konektivita je stavěna a připravena na mnohonásobně vyšší zátěž, než je ta aktuální a to zejména z důvodu dostatečné redundance a budoucího růstu. Pro představu informační systém společnosti běží na 64 jádrových serverech v sestavě 3 + 1, kde právě jeden server slouží pro real-time replikace a zálohování dat. Každý server pak disponuje 1GB internetovým připojením. Dále servery společnosti jsou geograficky decentralizovány (Vsetín, Praha) a datová centra vybavena automatickou klimatizací, automatickým hasícím systémem, zdvojenými napájecími okruhy atp. odpovídající třídě TIER III.

Hardware		Hodnocení	
1	Je možné současné HW vybavení označit za moderní a sledující současné trendy?	Ano	**5**
2	Přispívá HW pozitivně k rychlosti a použitelnosti informačního systému?	Ano	**5**
3	Nákup nového HW je posuzován s ohledem na ergonomii pro jeho uživatele?	Ano	**5**
4	Dá se připojení k počítačovým sítím označit za spolehlivé, dostatečně rychlé a vyhovující?	Ano	**5**
5	Jsou klíčové prvky HW dostatečně fyzicky chráněny před krádeží, požárem a povodní?	Ano	**5**
6	Je nové HW vybavení pořizováno po zvážení jeho kompatibility s existujícím HW vybavením a softwarem, který na něm bude provozován?	Ano	**5**
7	Současné HW umožňuje účinnou výměnu dat s odběrateli či dodavateli?	Ano	**5**
8	Je rychle dostupné záložní vybavení v případě výpadku klíčových HW prvků systému?	Spíše ano	**4**
9	Souhlasíte s výrokem, že současné HW vybavení bude do dvou let těžko použitelné?	Ne	**5**
10	Jsou poruchy HW vybavení na denním pořádku?	Ne	**5**
Celkem		**98%**	**49**

Tabulka 3 – Výsledek hodnocení hardware.

3.2.2 Software

Tuto část dotazníku je zaměřena na softwarové vybavení společnosti, respektive software využívaný k chodu IS. Hodnocený software byl/je vyvíjen samotnou společností na míru svým potřebám. Za jedinou negativní část lze označit obsažení všech nezbytných funkcí pro práci uživatelé ve zkoumaném software a to z toho důvodu, že všechny funkce, které společnost považuje za pracovně nezbytné, nebyly v době psaní této práce zcela dokončeny. Připomínám, že i předmětem této práce je návrh a implementace komponenty, která uživatelům přinese nové funkce potřebné pro svou práci.

Nápověda k software je vytvářena vlastní organizací a společnost zde nachází jisté mezery, kde je ovšem nutno zvážit poměr přínosu psaní dané dokumentace/nápovědy k vynaloženému času a úsilí. Nápověda externího software obvykle bývá zpracována velmi kvalitně. K otázce pořízení nových verzí software zástupce dodává, že nové verze jsou pořizovány především kvůli jejím novým vlastnostem.

Software		Hodnocení	
1	Poskytuje zkoumaný software všechny funkce nezbytné pro práci uživatelů?	Spíše ne	2
2	Je grafické členění plochy pro zadávání, editaci vstupních údajů přehledné a přispívá tak ke snadnosti práce se systémem?	Spíše ano	4
3	Jsou chybová, varovná hlášení či jiné nestandardní oznámení srozumitelná a poskytují na požádání i bližší vysvětlení vzniklé situace?	Ano	5
4	Rychlost zpracování úkolů jako tisky, dotazy, vyhledávání se jeví jako dostatečně rychlé?	Spíše ano	4
5	Platí, že koncoví uživatelé nesmějí poskytovat podněty pro případné úpravy SW, nové nastavení nebo pořízení nových verzí software?	Ne	5
6	Je nápověda k softwaru srozumitelná a přehledná?	Částečně	3
7	Má zkoumaný informační systém jednotné ovládání obrazovek, menu, sestav a nápovědy?	Ano	5
8	Jsou při pořízení nových verzí SW využívány jejich nové vlastnosti?	Ano	5
9	Je pravda, že snadnost používání softwaru koncovými uživateli nehraje roli při jeho pořízení nebo vývoji?	Ne	5
10	Existují pravidelné nebo nahodilé kontroly sloužící ke zjištění abnormalit ve využívání systému, jeho nesprávného užívání či zneužívání?	Ano	5
Celkem		**86%**	**43**

Tabulka 4 – Výsledek hodnocení software.

3.2.3 Orgware

Hodnocení pravidel pro provoz informačního systému a doporučené pracovní postupy odhalily několik drobných nedostatků. K prvnímu bodu - neexistuje téměř žádná směrnice pro zotavení IS z havarijní situace. Toto zástupci společnosti odůvodňují argumentem, že k situaci, která by se dala označit termínem „havarijní“ dojít vůbec nesmí – všechna kritická zařízení jsou replikována a v případě problémů je spuštěno záložní zařízení.

Pravidla pro bezpečnost IS pro nakládání s dokumenty zástupci společnosti nepovažují vzhledem k velikosti společnosti za kritické. (Otázka č. 3. se týká firem s řádově alespoň desítkami pracovníků.) Podobně je to také ohledně dozírání managementu na dodržování pravidel bezpečnosti a provozu IS.

Pravidla a politika bezpečnosti IS existují, ale pouze nepsanou formou a aktualizována jsou pouze s ohledem na aktuální stav SW a HW.

Orgware		Hodnocení	
1	Existují postupy či směrnice pro zotavení IS z nestandardních a havarijních situací a jsou tyto dokumenty dostatečně známé uživatelům?	Spíše ne	2
2	Existují doporučené pracovní postupy a procedury běžného provozu pro koncové uživatele a jsou udržovány v aktuálním stavu?	Spíše ano	4
3	Existují pravidla pro bezpečnost IS a obsahují i ustanovení pro nakládání s dokumenty či přílohami e-mailů získaných z Internetu?	Ne	1
4	Je pravda, že management příliš nedozírá na dodržování pravidel bezpečnosti a provozu IS?	Spíše ano	2
5	Má každý pracovník jasně určeno, s jakými úlohami smí pracovat a kdy?	Spíše ano	4
6	Provádějí jakékoliv rozsáhlejší instalace, změny nastavení, připojení nové techniky pověřené osoby, nikoliv uživatelé?	Ano	5
7	Jsou ošetřeny odchody zaměstnanců a ukončení platností jejich přístupových práv?	Ano	5
8	Existují pravidla nebo politika bezpečnosti IS a jsou tyto pravidelně aktualizovány?	Spíše ne	2
9	Umožňuje informační systém efektivní výměnu informací mezi uživateli IS v podniku?	Spíše ano	4
10	Platí, že pravidla pro provoz a bezpečnost IS jsou jasná a logická?	Ano	5
Celkem		**68%**	**34**

Tabulka 5 – Výsledek hodnocení orgware.

3.2.4 Peopleware

V otázkách vzdělávání a informovanosti pracovníků vzhledem k informačnímu systému společnost připouští určitý nedostatek spočívající v neexistenci dokumentů, které by byly za účelem školení nového personálu k dispozici. Zástupci firmy považují školení za důležitý informaticko-personální aspekt a přiznávají jeho nedostatečnou úroveň.

V otázce zastupitelnosti koncových uživatelů, kteří jsou pro chod systému klíčoví, společnost poznamenává závislost na konkrétní roli – někteří klíčoví pracovníci v určitém oboru své činnosti zastupitelní jsou, ostatní pak nikoliv (jsou pro firmu zcela klíčový a obvykle jsou to přímo jednatelé firmy).

Dokumentace běžných postupů práce s IS obvykle koncovým uživatelům dosažitelná není, protože tato dokumentace není úplná. Vedení firmy podporuje učení koncových uživatelů za účelem zvýšení efektivnosti fungování IS a to formou konzultací.

Peopleware		Hodnocení	
1	Je každý pracovník zaškolen na úlohy, které má s informačním systémem provádět?	Spíše ano	**4**
2	Jsou dostupná školení nových pracovníku o používaných informačních systémech, pravidlech provozu a bezpečnosti IS?	Ne	**1**
3	Je pravda, že stávající zaměstnance není třeba školit na nové funkce IS a že školení není dostupné?	Ne	**5**
4	Existuje zastupitelnost koncových uživatelů, kteří jsou klíčoví pro chod systému a jeho klíčové výstupy?	Částečně	**3**
5	Je dokumentace běžných postupů práce s IS jednoduše dosažitelná pro koncové uživatele?	Spíše ne	**2**
6	Je si management vědom vlivu firemní kultury na způsob práce koncových uživatelů s informačním systémem?	Ano	**5**
7	Jsou dostupná místa uvnitř firmy nebo u externího dodavatele, kam se mohou uživatelé obracet se žádostí o pomoc či konzultaci ohledně IS? (tato místa jsou označována dále jako informační centra)	Ano	**5**
8	Řeší informační centra z předchozího bodu podněty uživatelů obvykle v dostatečné míře a včas?	Ano	**5**
9	Je pravda, že informační centra především „hasí“ palčivé problémy a nemají důvod se snažit o dlouhodobé zlepšení chodu IS?	Spíše ne	**4**
10	Podporuje vedení firmy učení koncových uživatelů a jejich školení za účelem zvýšení efektivnosti fungování IS?	Ano	**5**
Celkem		**78%**	**39**

Tabulka 6 – Výsledek hodnocení peopleware.

3.2.5 Dataware

Odpovědnost za data, která pracovníci spravují, nelze u všech pozic zcela jednoznačně určit. Vyšším pracovním postem zároveň roste množství dat, za která je pracovník zodpovědný a která spravuje, přičemž tyto zodpovědnosti a práva pak jdou napříč systémem a často se kříží na úrovni jiných pracovních postů.

Pracovníci správy IS nemusejí pravidelně manuálně provádět zálohy a management nemusí dozírat na dodržování pravidel zálohování (otázka č. 5.). Proces zálohování zástupci považují za dobře navržený a zcela automatizovaný proces.

Klíčová data v informačním systému (otázka č. 7.) je možno snadno obnovit z inkrementálních záloh. Pověření pracovníci přesně vědí, jak v tomto případě postupovat, nicméně fyzicky tyto plány sepsány nejsou. Za dostatečnou katalogizaci záloh (otázka č. 4.) společnost považuje geografickou oddělenost a dostatečnou replikovatelnost.

Dataware		Hodnocení	
1	Mají pracovníci jasně vymezenou odpovědnost za data, která spravují? Tedy platí zásada, že určitá data smí měnit jen určitý pracovník?	Částečně	**3**
2	Mají pracovníci určeno, kdy musí jaká data zavést do informačního systému a kdy je musí aktualizovat?	Ano	**5**
3	Platí, že uživatelům chybí z informačního systému data pro jejich rozhodování?	Částečně	**3**
4	Získávají koncoví uživatelé nadbytečná nebo nepřesná data?	Ne	**5**
5	Musí pracovníci správy IS pravidelně provádět zálohování dat a dozírá management na dodržování pravidel zálohování?	Automaticky	**5**
6	Uznává management důležitý význam koncových uživatelů pro integritu a správnost zpracování dat?	Ano	**5**
7	Existují podrobné plány pro obnovu klíčových dat v informačním systému?	Spíše ano	**4**
8	Jsou média se zálohami dostatečně katalogizována a chráněna před zneužitím, krádeží či živelnou pohromou?	Spíše ano	**4**
9	Je bezpečnost dat zvažována a řízena i pro hrozby z Internetu nebo jiných počítačových sítí?	Ano	**5**
10	Mají pracovníci určeno, s jakými daty smí pracovat a s jakým oprávněním? Platí tedy zásada, že nikdo nesmí získat přístup k datům, která nepotřebuje pro svou práci?	Částečně	**3**
Celkem		**84%**	**42**

Tabulka 7 – Výsledek hodnocení dataware.

3.2.6 Customers

Metriky cílů zkoumaného informačního systému směrem k jeho zákazníkům (otázka č. 2.) jsou v rámci společnosti spíše kvalitativního než kvantitativního charakteru.

Společnost přijímá od svých uživatelů a zákazníků odezvy a řídí se jimi (otázka č. 3.). Tyto odezvy jsou zpracovány a zkoumány ihned po jejich přijetí.

V otázce řízení integrace zkoumaného IS spolu s dalšími IS podniku (otázka č. 9.) společnost dodává, že integrace je velmi důležitá a je jedním z hlavních cílů jejího snažení. (Integrací IS společnosti a ekonomickým softwarem se zabývá tato práce.)

Customers		Hodnocení	
1	Jsou jasně stanoveny základní cíle zkoumaného informačního systému směrem k jeho zákazníkům?	Ano	5
2	Existují metriky cílů uvedených v předchozím bodu a jsou dostatečně vyhodnocovány?	Spíše ne	2
3	Je pravidelně zkoumáno, jaké přínosy od informačního systému jeho zákazníci očekávají?	Ano	5
4	Je pravda, že názory zákazníků IS na zlepšení, změnu či úpravu informačního systému nejsou pro podnik důležité?	Ne	5
5	Jsou data o zákaznících IS, jejich požadavcích, operacích, atd. ukládány v informačním systému centrálně (tj. nejsou ukládány vícekrát nebo jinak nekonzistentně)?	Spíše ano	4
6	Přispívá současné hardwarové a softwarové vybavení k dostatečně rychlým odezvám na požadavky zákazníků IS?	Ano	5
7	Je forma výstupů z informačních systémů volena tak, aby umožňovala jejich snadné využití zákazníkem IS?	Ano	5
8	Ošetřují pravidla provozu nakládání s citlivými či obchodně cennými daty o zákaznících IS?	Ano	5
9	Je řízena integrace zkoumaného informačního systému firmy spolu s dalšími IS podniku, které poskytují výstupy pro dané zákazníky?	Spíše ano	4
10	Mohou zákazníci získávat ze zkoumaného IS výstupy pomocí různých komunikačních kanálů, které si zvolí?	Spíše ano	4
Celkem		**88%**	**44**

Tabulka 8 – Výsledky hodnocení customers.

3.2.7 Suppliers

Požadavky, kladené na dodavatele (bod 1.) musí být z povahy prací, které jsou zajišťovány externě, zcela jasně specifikovány. Zde se jedná především o:

1. vývoj aplikací pro klientské platformy (iOS, SmartTv, atp.),
2. právní služby,
3. dodavatele HW (set-top boxy),
4. konektivita (Freetel, Sychrovnet).

Kontroly informací od dodavatelů jsou v pravidlech provozu (otázka č. 4.) definovány pouze částečně v rámci aplikace, která tato data zpracovává.

Suppliers		Hodnocení	
1	Jsou jasně stanoveny základní požadavky kladené na dodavatele, které jsou nezbytné pro plnění definovaných cílů zkoumaného informačního systému?	Ano	5
2	Existují metriky hodnocení výše zmíněných požadavků a jsou dostatečně vyhodnocovány?	Ano	5
3	Je forma vstupů do zkoumaného IS od dodavatelů volena tak, aby umožňovala jejich snadné převzetí a využití zkoumaným IS?	Ano	5
4	Jsou v pravidlech provozu definovány kontroly informací od dodavatelů?	Spíše ne	2
5	Jsou požadavky na dodavatele ve vztahu ke vstupům do zkoumanému IS formulovány tak, aby byla jasně určená požadovaná podrobnost předávaných informací?	Spíše ano	4
6	Jsou požadavky na dodavatele ve vztahu ke vstupům do zkoumanému IS formulovány také s jasným určením požadované včasnosti jejich dodávání?	Ano	5
7	Zvažuje firma možnost účelného přizpůsobení či nastavení zkoumaného IS dle návrhů dodavatelů za účelem efektivnější výměny informací?	Spíše ano	4
8	Je forma výstupů ze zkoumaného IS pro dodavatele řízena s ohledem na efektivní komunikaci s dodavateli?	Spíše ano	4
9	Je pravda, že výstupy z IS pro dodavatele nejsou řízeny s ohledem na včasnost jejich předání?	Ne	5
10	Přispívá zkoumaný informační systém ke snadnosti a efektivnosti komunikace s dodavateli?	Ano	5
Celkem		**88%**	**44**

Tabulka 9 – Výsledky hodnocení suppliers.

3.2.8 Management IS

Management IS společnosti tvoří pracovníci s dlouholetou praxí a bohatými zkušenostmi v oboru informačních technologií. Níže uvedené otázky týkající se managementu IS pak tito lidé považují za naprostou samozřejmost. Investice do IS z ekonomického hlediska (bod 6.) není potřeba obhajovat, protože společnost považuje svůj technologický rozvoj za velmi důležitou konkurenční výhodu a schopnost na trhu uspět.

Management IS		Hodnocení	
1	Trvají manageři na dodržování pravidel stanovených pro informační systém?	Ano	**5**
2	Provádí řízení rozvoje a provozu informačních systémů osoba, která této oblastí rozumí?	Ano	**5**
3	Je rozvoj IS formulován také ve střednědobé či dlouhodobé perspektivě formou informační strategie vzhledem k cílům firmy?	Spíše ano	**4**
4	Je v plánech rozvoje informačních systémů zahrnut případný růst firmy a rozvoj jejích informačních potřeb?	Ano	**5**
5	Platí, že plány rozvoje IS neexistují nebo v nich nejsou stanoveny možnosti kontroly jejich plnění?	Spíše ne	**4**
6	Je při plánech rozvoje informačního systému, pořizování IS provedeno obhájení dané investice z ekonomického hlediska?	Ne	**1**
7	Považuje management informačních systémů koncové uživatele za faktor s vysokou důležitostí pro úspěšných chod informačních systémů?	Ano	**5**
8	Usiluje management IS soustavně o zlepšení efektivnosti chodu zkoumaného informačního systému?	Ano	**5**
9	Vnímá obecný management informační systém firmy nejen jako výdaje, ale také jako potenciál případného růstu firmy?	Ano	**5**
10	Podporuje obecný management firmy rozvoj informačních systémů, který je odůvodněný přispěním IS k dosažení podnikových cílů?	Ano	**5**
Celkem		**88%**	**44**

Tabulka 10 – Výsledky hodnocení managementu IS.

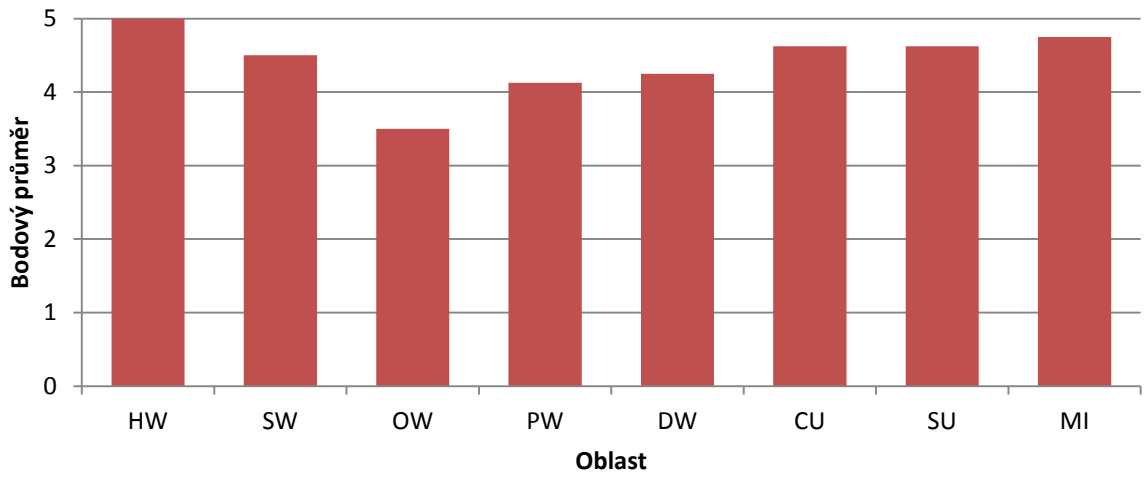
3.3 Výsledky hodnocení IS

Následující Tabulka 11 znázorňuje souhrn výsledků a vyhodnocení analýzy současného stavu informačního systému metodou HOS 8. (Postup je popsán v teoretické části.)

	HW	SW	OW	PW	DW	CU	SU	MI
Otázka 1	5	2	2	4	3	5	5	5
Otázka 2	5	4	4	1	5	2	5	5
Otázka 3	5	5	1	5	3	5	5	4
Otázka 4	5	4	2	3	5	5	2	5
Otázka 5	5	5	4	2	5	4	4	4
Otázka 6	5	3	5	5	5	5	5	1
Otázka 7	5	5	5	5	4	5	4	5
Otázka 8	4	5	2	5	4	5	4	5
Otázka 9	5	5	4	4	5	4	5	5
Otázka 10	5	5	5	5	3	4	5	5
Průměr	**4,9**	**4,3**	**3,4**	**3,9**	**4,2**	**4,4**	**4,4**	**4,4**
Min	**4**	**2**	**1**	**1**	**3**	**2**	**2**	**1**
Max	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
Přesné u_i	**5,0**	**4,5**	**3,5**	**4,1**	**4,3**	**4,6**	**4,6**	**4,8**
u_i	**5**	**5**	**4**	**4**	**4**	**5**	**5**	**5**

Tabulka 11 – Souhrn výsledků metody HOS 8 a hodnoty stavů jednotlivých oblastí.

Všimněme si, že podnik dosáhl ve všech oblastech velmi dobrých výsledků. Z následujícího grafu je patrný vyrovnaný průměrný výsledek jednotlivých oblastí, který zároveň neklesl pod hodnotu 3,4. Téměř 59 % otázek bylo zodpovězeno zcela pozitivně a jen 4 % negativně.



Graf 3 – Průměrný bodový zisk v jednotlivých oblastech.

Jako nejslabší oblast podnikového IS se jeví Orgware. Tento výsledek odůvodňuji poměrně krátkou působností společnosti, kdy patřičné postupy, pravidla, směrnice či politika bezpečnosti IS doposud nebyly jasně stanoveny, případně nezískaly formu konkrétních dokumentů, jejichž vypracování vzhledem k nízkému počtu pracovníků společnost nepovažuje v tuto chvíli za prioritní.

3.3.1 Vyváženost IS

Vyváženost IS podle metodiky HOS 8 se určuje, zda je úroveň jednotlivých oblastí na přibližně stejné úrovni. Následující Tabulka 12 uvádí nominální hodnoty jednotlivých oblastí (dle vzorce, který je uveden v teoretické části) – u_i a rozdíl mezi nominální hodnotou úrovně každé oblasti a hodnotou úrovně celého IS (ten je stanoven jako minimum hodnot dílčích oblastí a nazývá se *vyvážený systém*) – $(u_i - u)$.

$$\boldsymbol{u} = \min(5;\ 5;\ 4;\ 4;\ 4;\ 5;\ 5;\ 5) = \mathbf{4}$$

	HW	**SW**	**OW**	**PW**	**DW**	**CU**	**SU**	**MI**
u_i	5	5	4	4	4	5	5	5
$u_i - u$	1	1	0	0	0	1	1	1

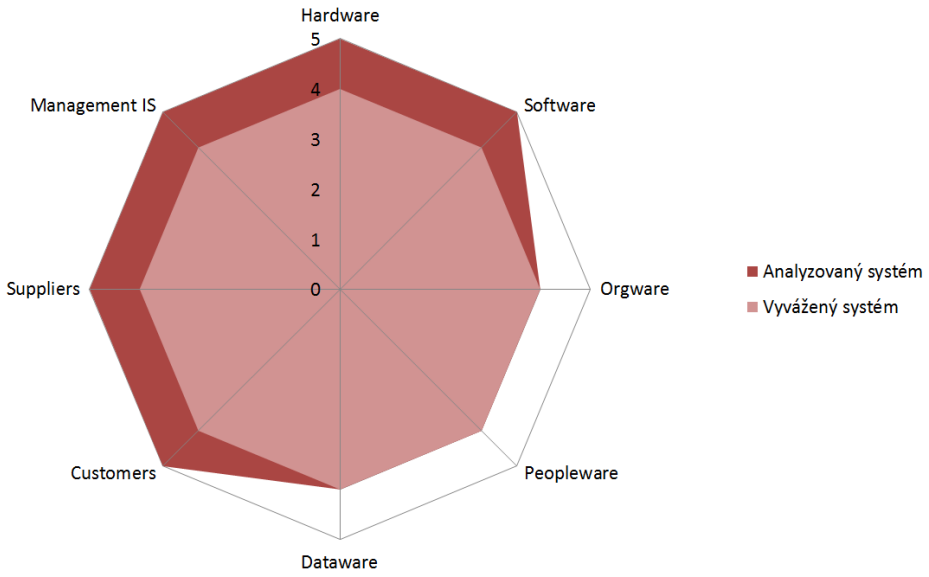
Tabulka 12 – Nominální hodnoty jednotlivých oblastí.

Připomínám a zároveň dosazuji do podmínek pro určení vyváženosti IS. První podmínka vyjadřuje, že rozdíly mezi hodnotami jednotlivých oblastí a hodnotou vyváženého systému se mají rovnat nebo být menší než 1 ($u_i - u \leq 1$). Výsledek je z druhého řádku tabulky zřejmý a dle této podmínky konstatuji, že **IS je vyvážený**.

Druhá podmínka stanovuje, že maximálně 3 hodnoty smějí být vyšší (nižší být nemohou) než je hodnota vyváženého systému. Dosazením do vzorce získáváme:

$$\sum_{i=1}^{8} (u_i - 4) = \mathbf{5}$$

Vyváženost IS druhé podmínce **neodpovídá,** což ale přináší i pozitivní význam, ze kterého plyne, že více než 3 oblasti jsou na lepší úrovni, než je úroveň vyváženého systému. Graficky je toto interpretováno následujícím pavučinovým grafem (Graf 4).



Graf 4 – Celková vyváženost zkoumaného IS.

Závěrem – předmětná firma se řadí mezi organizace s velmi vysokou důležitostí informačního systému *(v = 1)*, to znamená, že význam informačního systému pro chod podniku je klíčový a jeho krátkodobý výpadek má za následek značné škody. Zkoumané informační systémy zařazené do tohoto stupně by měly mít úroveň stavu IS *u = 4*. A této podmínce společnost **vyhovuje**.

3.4 Analýza změn

Závěrem analytické části této práce přichází na řadu analýza konkrétních změn, ke kterým má dojít a které budou předmětem implementačních prací.

Předmětem změn bude část informačního systému společnosti, který si sama vyvíjí, a ekonomický software (ES[14]), který je využíván k plnění veškeré účetní agendy. Údaje, které jsou uloženy v informačním systému společnosti (IS[15]), nejčastěji fakturační údaje, musí být jednou měsíčně ručně přeneseny a zadány do ES, který informace a data zpracuje a výstupy, nejčastěji ve formě faktur, jsou znovu manuálně přenášeny do IS. Toto s sebou kromě vysoké pracnosti přináší také riziko chybovosti. Zde je navíc mezi přenášením samotných dat také zapotřebí manuálně provádět výpočty – výpočty fakturovaných částek na základě velmi variabilně nastavených smluv s jednotlivými partnery.

[14] Pro termín „ekonomický software“, případně „ekonomický systém“ je dále použita zkratka „ES“.
[15] Pro termín „informační systém společnosti“ je dále použita zkratka „IS“.

Vzhledem k neustále narůstajícímu počtu klientů není tento stav pro společnost již dále udržitelný a je zapotřebí celý tento proces automatizovat. S nastalou situací společnost již dříve kalkulovala a tímto byl již při zahájení její činnosti zvolen ES, který podporuje automatizované zpracování či dávkové příkazy prostřednictvím vzdáleného rozhraní (API).

Konkrétním cílem změn je ve zkratce předávání fakturačních údajů do ES, kde jsou tato data zpracována, uložena a vygenerovány výstupy. Výstupem ES jsou kupříkladu faktury, které musejí být zpětně přístupné prostřednictvím podnikového IS zákazníkům. Zákazníci ve svém klientském rozhraní mají mít přehled o všech fakturách včetně možnosti prohlížet platební údaje a zobrazovat faktury. Podklady pro fakturu jako takovou se dále sestavují z několika dalších dílčích objektů, které musejí být bezpodmínečně zpracovány také. Tímto vzniká jeden celistvý projekt, jehož vyřešení a implementaci si tato práce klade za cíl.

3.4.1 Řízení kontaktů

První dílčí změna se bude zabývat managementem kontaktů, protože právě synchronizace kontaktních a fakturačních údajů mezi oběma systémy je nezbytná pro budoucí řízení vyúčtování, fakturace a plateb.

V současnosti jsou v IS uloženy pouze záznamy partnerů pouze v takovém rozsahu, jaký je pro jeho fungování zapotřebí a údaje slouží pouze pro základní identifikaci partnera (klienta).

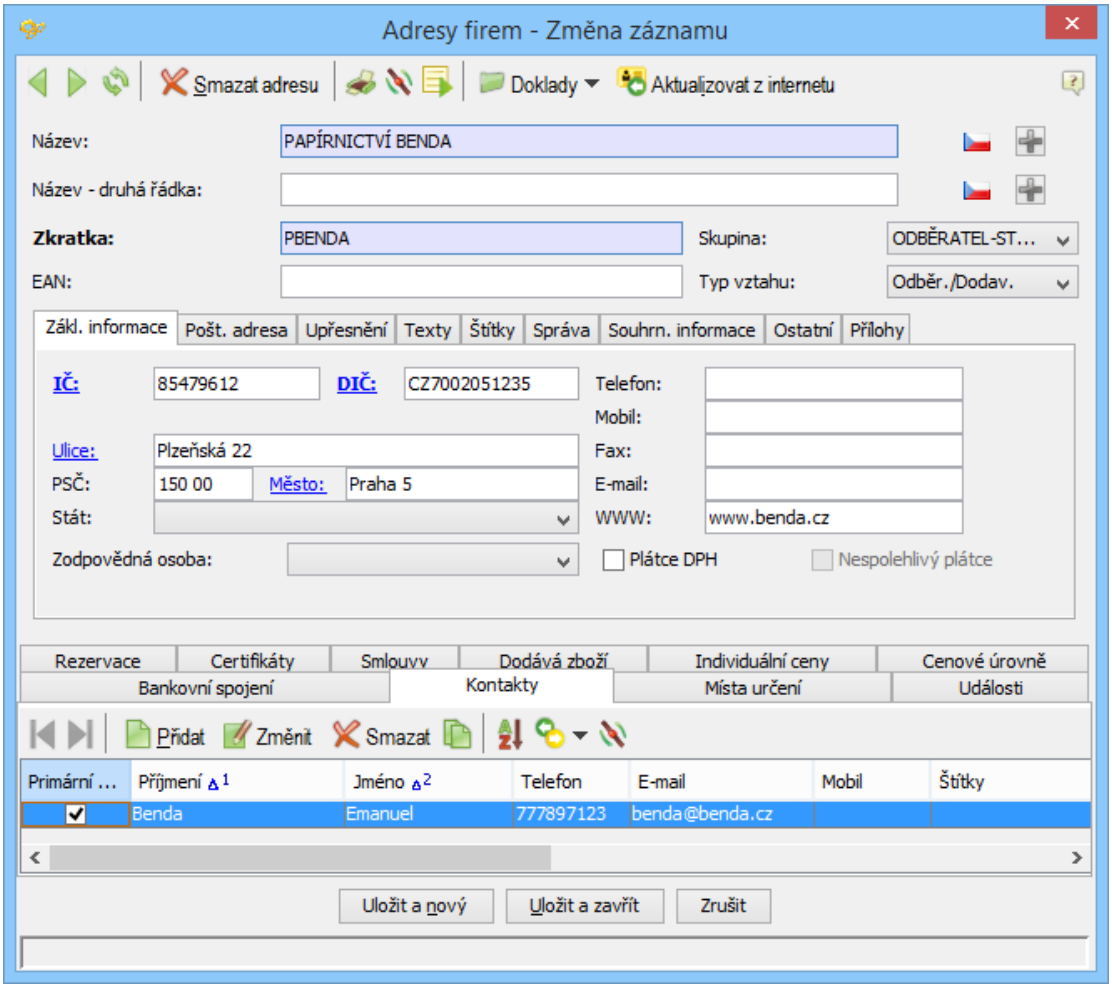
Údaje o partnerech se v databázi IS skládají z následujících položek:

Atribut	Datový typ	Poznámka	Popis
Id	INT	PK	jednoznačný identifikátor, PK
Name	VARCHAR(150)		jméno/název partnera
Provider	INT	FK provider	identifikace providera
Api	BOOLEAN		povolení využívat interní API
Adress	VARCHAR(1024)		fakturační adresa
Login	VARCHAR(64)	unique	přihlašovací jméno
Password	VARCHAR(64)		hash přístupového hesla
MailAdress	MEDIUMTEXT		korespondenční adresa
Ready	BOOLEAN		schválení uživatele
Website	VARCHAR(255)		url
UserPriceList	INT	FK userPriceList	ceník partnera
Locale	CHAR(2)		jazyk uživatele

Tabulka 13 – Atributy databázové tabulky Partners.

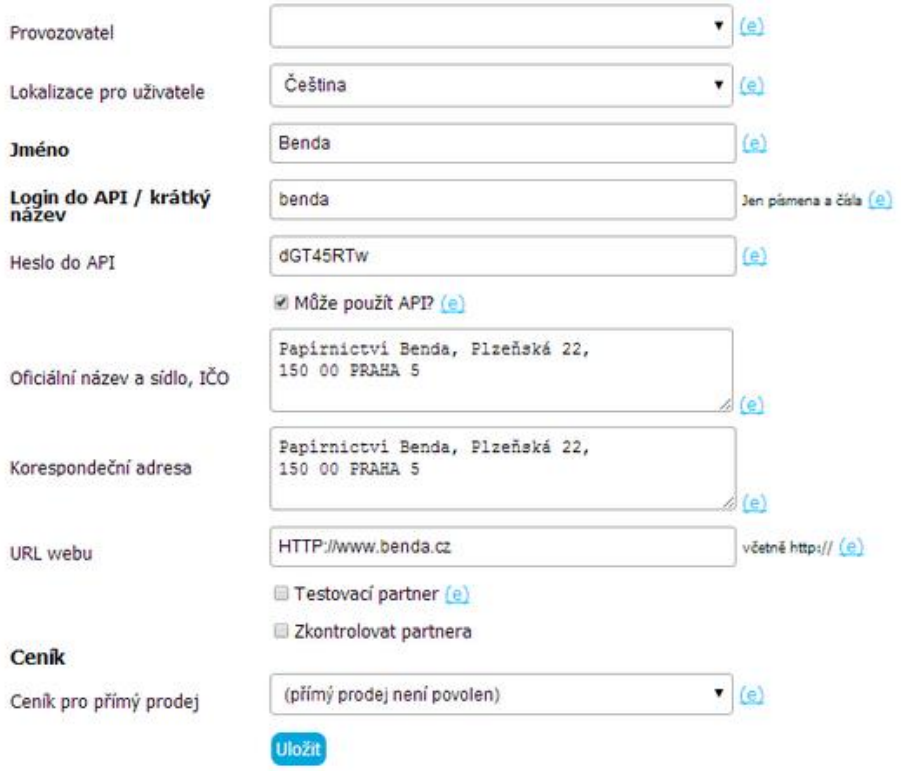
Z tabulky (Tabulka 13) je zřejmé, že fakturační a korespondenční adresy jsou uloženy ve dvou atributech (*Adress* a *MailAdress*) ve formě prostého textu bez standardizace. V praxi jsou tyto záznamy udávány tak, jak jej obchodní oddělení do IS zadá (neplatí žádná formátovací pravidla). Následně údaje zpracovává fakturantka ve chvíli, kdy je zapotřebí danému partnerovi vystavit fakturu. Fakturantka podle uvedených údajů buďto klienta v ES vyhledá a použije, případně vytvoří nový kontakt a zadá předmětné údaje.

Zde vzniká velký problém v případě, že je kontakt v jedné z databází aktualizován. Pokud je kontakt aktualizován v interní databázi IS, pak fakturantka toto nemusí postřehnout (a tento fakt zároveň není nijak monitorován) a použije, může pro fakturaci použít neplatné údaje. Na druhou stranu změnou údajů v ES se změna nemůže do IS přenést jinak, než manuální editací záznamu partnera. Zde pak samozřejmě platí pravidlo, že to, co se neděje automatizovaně, se neděje vůbec.



Obrázek 2 – Práce s kontaktem v ES (Flexibee).

Obrázek 2 a Obrázek 3 demonstrují přidání či editaci kontaktu v ES a interním IS.



Obrázek 3 – Část formuláře pro vložení/editaci partnera v IS.

Situaci ohledně kontaktů bude dále komplikovat skutečnost, že každý partner je přiřazen providerovi (poskytovatel) a zde platí, že více partnerů může být sdruženo pod jedním providerem. Důležitou vlastností providera je vždy působnost v rámci jednoho státu, což mimo jiné znamená přiřazení k určitému účetnictví. Společnost pro každý stát, a v době psaní této práce se usilovně pracuje na slovenské expanzi, musí vést oddělené účetnictví. Toto se bude projevovat na zařazování faktur do jednotlivých účetnictví dle nastavení providera.

3.4.2 Řízení fakturace

Situace ohledně samotné fakturace je pak velmi podobná té, výše podobné situaci s kontaktními údaji. Pro lepší orientaci uvádím celý **proces fakturace** v bodech:

1. Obchodní oddělení vytvoří smlouvu s providerem a zavede providera do IS.
2. V rámci providera je do IS zanesen konkrétní partner (odebírající službu).
3. Partner odebírá služby dle smlouvy a ta je mu účtována dle smluvního ceníku.
4. Využité služby jsou partnerovi vyúčtovány a fakturovány měsíčně.
5. Měsíčně je partnerovi vytvořena statistika využitých služeb.

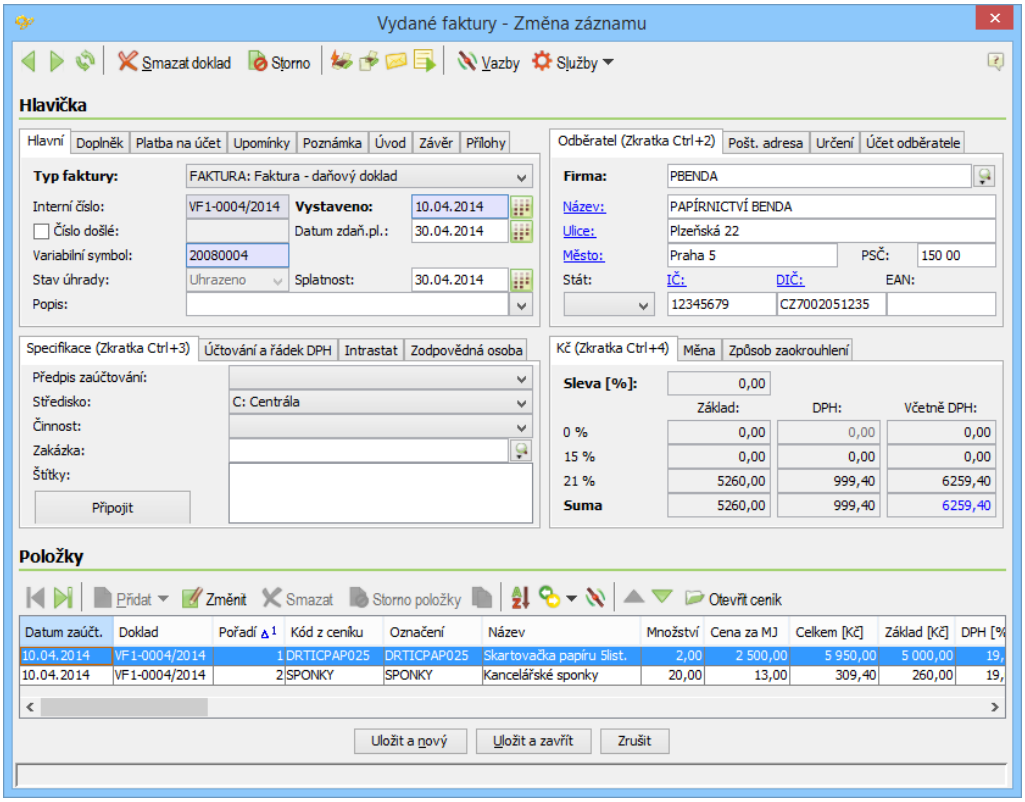
6. Statistika je porovnána s individuálním ceníkem a vypočítána dlužná částka.
7. Fakturantka zjistí dlužnou částku (prostřednictvím podkladů pro vyúčtování, které IS vygeneruje). Provede kontrolu a údaje zadá do ES.
8. Faktura je zaslána partnerovi (elektronicky nebo korespondenčně).
9. Platby partnerů jsou zpracovávány manuálně a každá platba musí být zároveň zadána také do ES.

Nedostatky tohoto systému:

1. Partner nemá možnost sledovat své vyúčtování a stav faktur online.
2. Řízení pohledávek probíhá výhradně prostřednictvím ES.
3. Platby jsou zpracovány se zpožděním a může docházet k chybám (lidský faktor).
4. Údaje nejsou uloženy jednotně a zcela chybí jejich integrace.

Kompletní zajištění fakturačních a platebních úkonů je zajištěno výhradně prostřednictvím ES. Toto se při rostoucím počtu klientů společnosti začíná být administrativně velmi náročné a pracné, proto je v této fázi vývoje společnosti důležitá automatizace této agendy.

Následující obrázek demonstruje složitost manuálního zadání každé vydané faktury do ES.

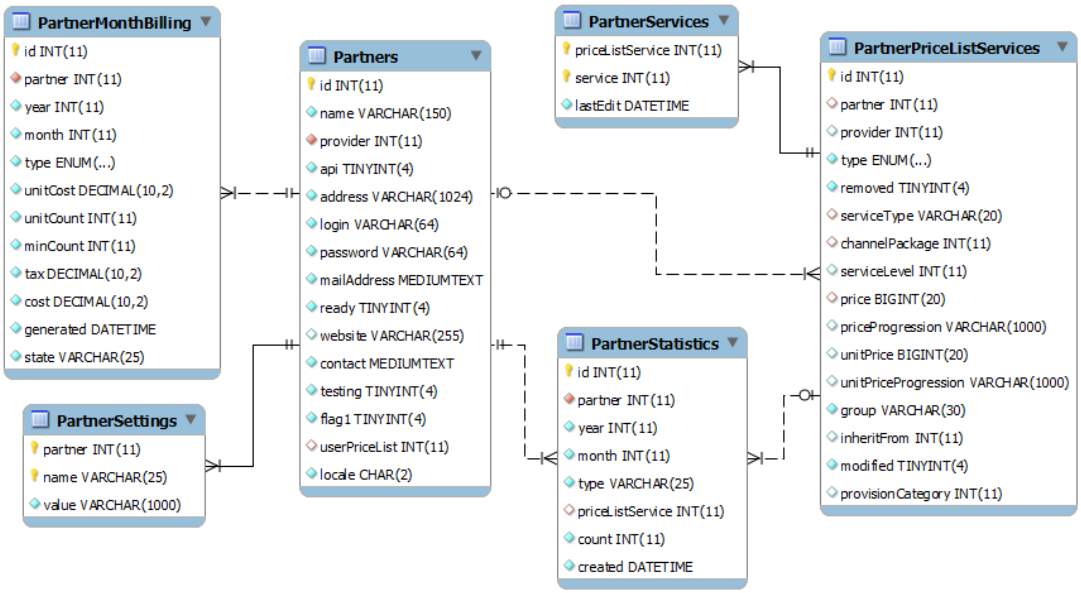


Obrázek 4 – Manuální vytvoření vydané faktury v ES.

3.4.3 Podklady účtování

Za účelem podpory fakturace je v informačním systému integrováno generování podkladů pro vyúčtování. Z technického pohledu se jedná o zjištění využitých služeb a výpočet cílové částky (dlužná měsíční částka), kterou ovlivňuje řada faktorů. Pro představu složitosti výpočtu fakturované částky uvádím základní přehled proměnných, které do výpočtů vstupují:

1. Využití **základní služby** (dané smlouvou) partnerem.
    a. Statistika **využívání** dané služby (objem jednotek).
        - **Fixní** počet jednotek (tarif).
        - **Variabilní** počet jednotek (nad rámec tarifu).
    b. **Progrese** – diference ceny jednotlivých služeb v závislosti na objemu jejich využití. Toto je navíc s každým partnerem nastaveno smluvně a odlišně na základě vyjednání obchodními zástupci.
        - **Slevy** na základě dosažených úrovní (objemu) využití služeb.
    c. **Daně a poplatky** dle zařazení do účetnictví (stát).
        - **Poplatek ochrany autorských práv**.
        - **Daň z přidané hodnoty**.
        - **Měna** a měnové přepočty.
2. Využití **ostatních služeb**, kterých může klient využít více v různém objemu, přičemž ceník každé služby je stanoven individuálně. (Opakování bodu 1. s odlišnými cenami.)



Obrázek 5 – Entity udržující podklady pro výpočty vyúčtování a následnou fakturaci.

4 VLASTNÍ NÁVRHY ŘEŠENÍ

V této kapitole představuji návrh řešení. Odůvodněním volby ekonomického systému v první části přejdu na popis návrhu aplikace a to především databázový a funkční model. V druhé části následuje popis implementace – logické rozdělení, kódování a zapojení jednotlivých tříd a jejich využití.

4.1 Výběr ekonomického systému

Přestože cílem této práce není výběr ekonomického software, považuji za vhodné argumenty výběru uvést. Společnost ihned po svém založení přešla do technologického inkubátoru JIC a v krok přispěl ke správnému řešení realizace účetních procesů. Společnost zvolila ekonomický systém FlexiBee a to z následujících důvodů:

API – pod těmito třemi písmeny je schován požadavek, který je pro společnost kritický, a který okamžitě vyřazuje 90 % účetních systémů na českém trhu. Jak již bylo uvedeno, tak cílem společnosti je maximalizace automatizace a to včetně (do jisté míry) některých cyklických účetních úkonů. API představuje možnost externího zadávání dotazů do účetního systému a tyto externí zásahy lze zajistit také programově.

Cloudové uložiště – všechna účetní data jsou uložena mimo systémy Sledovanitv.cz, to společnosti zásadním způsobem šetří práci ohledně skladování a údržby těchto dat. Data jsou zároveň dostupná odkudkoliv v rámci internetové sítě a za jejich údržbu odpovídá provozovatel.

Cena – Jihomoravské inovační centrum jako odběratel multilicence tohoto systému dosahuje na podstatně nižší ceny, než jaký je standardní ceník FlexiBee Systems.

Podpora – v rámci JIC pracují pracovníci vyškolení na používání tohoto software a pracovníci, kteří firmám a projektům účetnictví zpracovávají. V případě problémů je pak vždy na blízku odborná pomoc a firmy se tak mohou soustředit výhradně na své primární činnosti.

Působnost na Slovensku – klíčová je také schopnost systému zpracovávat účetnictví pro Slovenské společnosti, protože právě Slovensko je v době psaní této práce místem první expanze analyzované společnosti.

Poznámka: 15. 4. 2014 FlexiBee Systems oznámila spojení s ABRA SOFTWARE a ve společném prohlášení plánují expanzi na zahraniční trhy a řadu dalších vylepšení pro své klienty.

4.2 Synchronizace kontaktů

Společnost požaduje uložení kontaktních, především pak fakturačních údajů, v interní databázi IS a jejich oboustrannou synchronizaci s ES. Základní atributy pro kontakty jsou následující:

Atribut	Datový typ	Popis
Name	VARCHAR(255)	jméno či název organizace
Street	VARCHAR(255)	adresa sídla
City	VARCHAR(127)	město
Zip	VARCHAR(255)	poštovní směrovací číslo
Tel	VARCHAR(20)	primární telefonní kontakt
Mob	VARCHAR(20)	mobilní kontakt
Fax	VARCHAR(20)	číslo faxu
Email	VARCHAR(255)	emailový kontakt
CompanyNum	VARCHAR(20)	identifikační číslo organizace
vatNum	VARCHAR(20)	daňové identifikační číslo

Tabulka 14 – Datová struktura tabulky *PartnerContact* – základní údaje.

U každého kontaktu je zároveň důležité rozlišit, zda se kontaktní údaje shodují s fakturačními a v případě že ne, tak udržovat v databázi také aktuální fakturační údaje:

Atribut	Datový typ	Popis
PostIdentic	CHAR(5)	poštovní adresa identická s fakturační
InvName	VARCHAR(255)	název odběratele
InvStreet	VARCHAR(255)	fakturační adresa
InvCity	VARCHAR(127)	město
InvZip	VARCHAR(255)	poštovní směrovací číslo

Tabulka 15 – Datová struktura tabulky *PartnerContact* – fakturační údaje.

Základní atributy schématu relace jsou tímto zadány. Nyní vzniká otázka uložení. Buď je možno údaje uložit jakou součást tabulky *Partners*, nebo je umístit do samostatné relace a propojit prostřednictvím cizího klíče. V tomto případě se jako vhodnější jeví umístění do samostatné relace z hned několika důvodů:

1. možnost navázat na partnera více kontaktů (vztah 1:1 lze bez zásahu rozšířit na 1:N),
2. rozšířit relaci o další atributy aniž by „nabobtnávala“ tabulka partnerů (méně přenesených dat, větší přehlednost),
3. kontaktní údaj jako takový je samostatný objekt ať už ve slova-smyslu reálného světa tak implementačním a oddělení přispěje přehlednosti a rozšiřitelnosti.

K základním atributům nyní přidáme nezbytné rozšířené atributy, které ke svému správnému fungování bude využívat aplikace. Jedná se o následující:

Atribut	Datový typ	Vlastnost	Popis
`Id`	`INT`	`PK`	jednoznačný identifikátor, PK
`Partner`	`INT`	`FK partners`	cízí klíč tabulky Partners
`LastUpdate`	`TIMESTAMP`	`CURRENT_TIMESTAMP`	datum a čas poslední aktualizace v DB
`Code`	`CHAR(20)`	`unique`	unikátní zkratka partnera (klíč pro spárování s ES)
`RelationType`	`ENUM`		typ vztahu s kontaktem (dodavatel, odběratel)

Tabulka 16 – Datová struktura tabulky *PartnerContact* – rozšířené informace.

Atribut *LastUpdate* udržuje datum a čas poslední aktualizace záznamu. Tento údaj se v aplikaci stane základním kamenem pro vyhodnocení, ve které databázi (IS či ES) je záznam nový či zastaralý a podle toho budou data přenesena jedním, případně opačným směrem. Atribut *Code* slouží pro identifikaci záznamu v rámci ES. Tímto jednoznačným identifikátorem aplikace bude moci záznamy spárovat.

4.3 Fakturace

Faktura, jako objekt reálného světa, má několik zásadních vlastností, které je důležité vzít v potaz ještě dříve, než dojde k samotnému návrhu řešení databázového obrazu a aplikačního rozhraní.

1. Každá faktura musí být adresována odběrateli. Informace o daném odběrateli bude aplikace čerpat (pokud toto nebude stanoveno jinak) z objektu *PartnerContact*.
2. Změnou kontaktu daného partnera (editace objektu *PartnerContact*) se tyto změny nesmí projevit na již vydaných fakturách. Kontakt faktury s tímto stává samostatným objektem. (Dále již jen *InvoiceContact*).
3. Faktura může dle stavu zúčtování nabývat statusů:

   a. nezúčtovaná faktura – lze provádět změny faktury i položek faktury,
   b. zúčtovaná faktura – již nelze provádět žádné změny.
4. Každá faktura je přiřazena účetní jednotce (objekt *Accounting*). Zde platí, že každá jednotka může využívat a využívá různou instanci ekonomického software, případně vůbec služeb automatizace využívat nebude.
5. Fakturu je dle typu vyúčtování nutno rozlišit podle:
   a. váže se na konkrétní primární vyúčtování pro daného partnera za dané období,
   b. váže se na poskytnutí dalšího zboží či služeb mimo primární vyúčtování pro daného partnera za určené období,
   c. váže se na partnera, který je veden pouze v ES. Faktury tohoto typu budou v IS ignorovány.
6. Faktura smí obsahovat libovolné množství položek, přičemž i změna jediné položky je podnětná pro synchronizaci faktury jako celku. Položka faktury je samostatným objektem (*InvoiceItem*).
7. Každá faktura musí být před odesláním do ES zkontrolována fakturantkou a označena jako potvrzená.

Konkrétní atributy a jejich význam je uveden v následující tabulce.

Atribut	Datový typ	Vlastnost	Popis
Id	INT	PK	jednoznačný identifikátor, PK
Contact	INT	FK invoiceContact	cizí klíč tabulky InvoiceContact
Accounting	INT	FK accounting	cizí klíč tabulky Accounting
Flexibee	INT		cizí identifikátor ES
Status	CHAR(50)		stav faktury
InEvidence	TINYINT	unique	stav zaúčtování
LastUpdate	TIMESTAMP		datum a čas poslední aktualizace
Code	CHAR(20)		kód faktury např. VF1-0030/2014
Lock	CHAR(20)		stav uzamčení faktury
Vs	CHAR(30)		variabilní symbol
IssueDate	DATE		datum vydání
VatDate	DATE		datum zdanitelného plnění
PayMaxDate	DATE		datum splatnosti
PayDate	DATE		datum úhrady
Confirmed	TINYINT		stav potvrzení faktury v IS fakturantkou

Tabulka 17 – Datová struktura tabulky *Invoice*.

S relací *Invoice* souvisejí další objekty, které ukládají sekundární data a tato data jsou členěna v dalších relacích, jejichž popis nyní bude následovat.

4.3.1 Accountings

Je relací udržující nastavení účetnictví pro danou účetní jednotku. Již nyní společnost působí na dvou trzích – v ČR a na Slovensku. Pro obě země je zároveň využíván stejný ES, avšak každá účetní jednotka využívá samostatnou instanci (účet). Pro synchronizaci všech údajů mezi IS a ES musí aplikace správně nastavit účetnictví, se kterým se má dále pracovat. Tabulka *Accountings* udržuje nastavení jednotlivých účetních instancí tak, jak jej popisuje následující tabulka atributů (Tabulka 18).

Atribut	Datový typ	Popis
`Id`	`INT`	jednoznačný identifikátor, PK
`Name`	`VARCHAR(100)`	název účetní jednotky
`System`	`VARCHAR(50)`	ekonomický systém
`sHost`	`VARCHAR(128)`	ES host (url/IP)
`sCompany`	`VARCHAR(256)`	identifikátor společnosti
`sUser`	`VARCHAR(64)`	uživatelské jméno pro ES
`sPassword`	`VARCHAR(64)`	heslo pro ES
`Prescriptions`	`TEXT`	předpisy zúčtování (JSON)

Tabulka 18 – Atributy relace *Accountings*.

Cizí klíč *Acccountings* přebírá kromě relace *Invoice* také *Providers*, protože každý poskytovatel musí být zcela jednoznačně zařazen do právě jedné účetní jednotky, která mu využité služby účtuje.

4.3.2 Invoice Contact

Jak bylo uvedeno, každá faktura je jednoznačně přiřazena odběrateli, přičemž data odběratele se po zúčtování faktury již nesmějí změnit. Tímto vzniká potřeba samostatné relace, která bude udržovat fakturační data odběratele. Tabulka *InvoiceContact* je téměř kopií tabulky *PartnerContact* očištěnou o některé nesouvisející atributy. Zde je nutno podotknout, že záznam v tabulce *InvoiceContact* nemá po svém vytvoření (ať už je jeho zdroj z IS nebo ES) vztah se žádným záznamem v tabulce *InvoicePartner*. Struktura relace je zřejmá z výřezu databázového schématu, které je uvedeno na konci této kapitoly.

4.3.3 Partner Invoice

Je pomocnou relací mezi *Partners* a *Invoice*, která navíc udržuje další důležitá data o přiřazení dané faktury konkrétnímu odběrateli.

Každé vyúčtování je odběrateli generováno měsíčně. Relace *PartnerInvoice* ukládá údaje o roku a měsíci, pro který je vyúčtování určeno a zároveň typ faktury, protože je nutno uvažovat možnosti vystavení více faktur jednomu odběrateli v rámci jednoho měsíce a zde musí existovat možnost rozlišit standardní vyúčtování smluvních služeb od ostatních faktur (například odběr hardware). Atribut *type* může nyní (v rámci *ENUM*) nabývat hodnot *MonthServices* a *Other* (značící ostatní fakturaci).

Atribut	Datový typ	Vlastnost	Popis
Id	INT	PK	jednoznačný identifikátor, PK
Partner	INT	FK partners	cizí klíč tabulky Partners
Invoice	INT	FK invoice	cizí klíč tabulky Invoice
Year	SMALLINT		vyúčtování pro rok
Month	SMALLINT		vyúčtování pro měsíc v roce
Type	ENUM	unique	typ vyúčtování

Tabulka 19 – Atributy relace *PartnerInvoice*.

4.3.4 Invoice Items

Položka faktury je samostatným objektem a to také v ES. Položka faktury udržuje informace o fakturované službě či zboží, počtu jednotek, předpisu účtování a především peněžních hodnotách – cena za jednotku, daň z přidané hodnoty, zúčtovací měna. Tyto peněžní údaje jsou na základě své složitosti vyčleněny do samostatného objektu (*Money*) avšak jeho popis je již nad rámec této práce, princip fungování je ve zkratce popsán v části věnované implementaci řešení. Položka faktury tak udržuje pouze cizí klíč peněžní relace.

K položce faktury se dále vztahuje statistika (objekt *Statistics* udržující informace o využitých službách odběratele v daném období) která rozšiřuje informace fakturované položky na konkrétní využité služby. Fakturační model je uvažován tak, že ve standardním případě faktura obsahuje pouze jedinou položku, která je sumou dlužných částek za všechny využité služby vypočítané na základě statistiky za dané období.

Každá položka faktury udržuje informaci o předpisu zúčtování. Různé varianty těchto předpisů jsou uloženy ve složené (*array*) hodnotě atributu relace *Accounting*. Předpisy zúčtování jsou definovány v ES a prakticky odkazují na účty, do kterých má být položka účetně zařazena například „tržby za zboží“ či „tržby za služby“ a podobně.

Atribut	Datový typ	Vlastnost	Popis
Id	INT	PK	jednoznační identifikátor, PK
LastUpdate	TIMESTAMP	default CURRENT_TIMESTAMP	datum a čas poslední aktualizace záznamu v DB
Units	INT		počet jednotek
Statistics	INT	FK statistics	cizí klíč tabulky Statistics
Invoice	INT	FK invoice	cizí klíč tabulky Invoice
Money	BIGINT	FK money	cizí klíč tabulky Money
Description	TEXT		název či popis položky
Flexibee	INT		cizí identifikátor ES
Prescription	VARCHAR(60)		předpis zúčtování

Tabulka 20 – Datová struktura tabulky *InvoiceItem*.

4.4 Model

Následující ERR diagram (Diagram 12) představuje databázové entity, na kterých je implementační část této práce postavena. Jednotlivé atributy vycházejí z potřeb společnosti na uložení určitých dat a byly již dříve popsány.

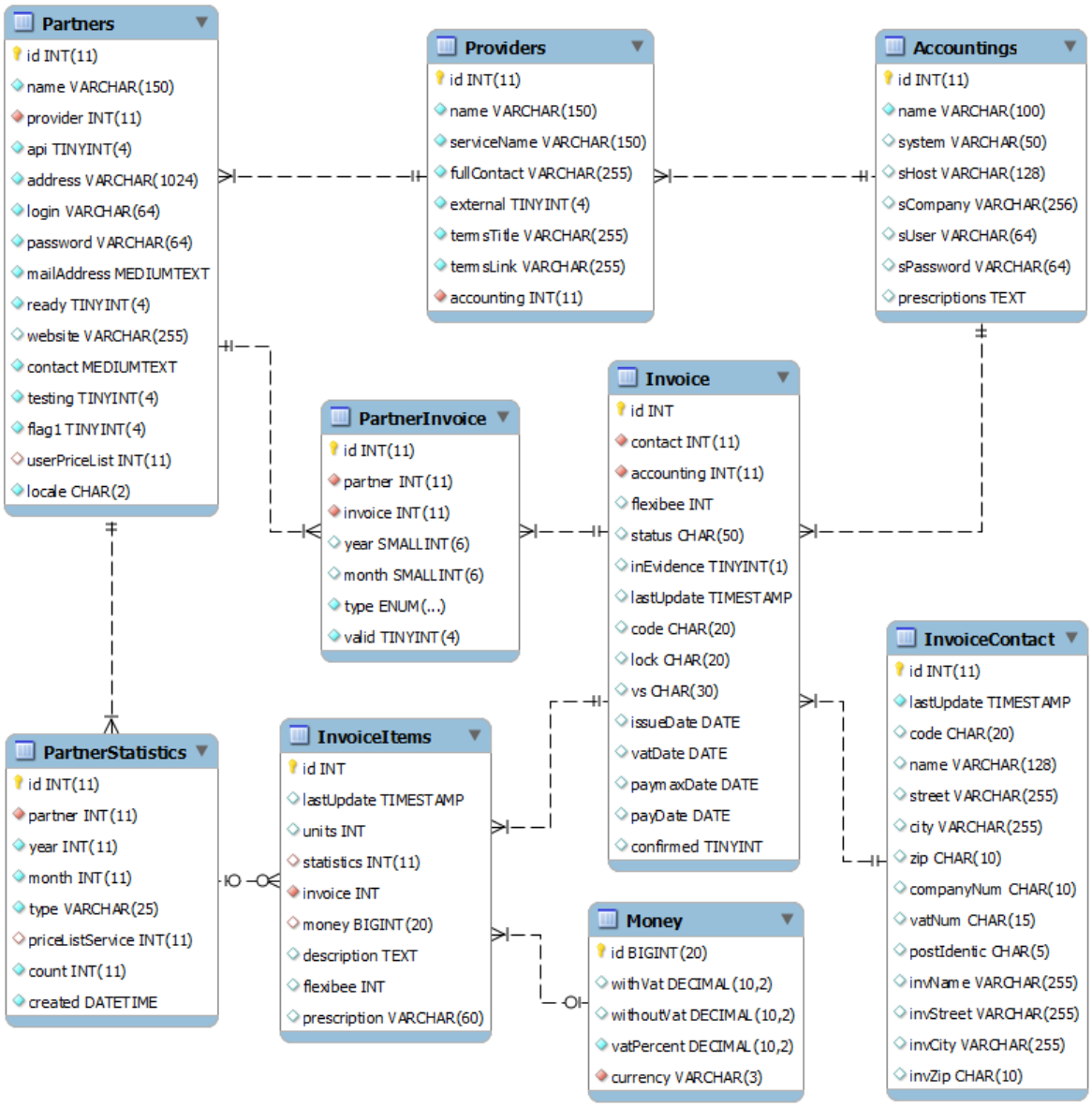


Diagram 12 – Část entit databázového modelu, se kterými bude pracovat implementační část. (Zdroj: vlastní tvorba)

Kompletní IS společnosti obsahuje téměř sto entity uložených v databázovém systému MariaDB. Práci s DB zajišťuje vrstva NotORM (součástí Nette) za využití InnoDB[16] enginu.

[16] Typ MySQL uložiště podporující práci s cizími klíči.

4.5 Implementace řešení

V této části práce si kladu za cíl popsat realizaci navrženého řešení formou programové implementace. Záměrem je napsání stručného průvodce principy aplikace spolu s ukázkami stěžejních částí programu (především pak zdrojového kódu). Objekty reálného světa, tak jak byly popsány v předchozí kapitole a převedeny na databázové entity budou nyní reprezentovány jako objekty programové, jejichž metody budou zastupovat funkce a stavy, jichž je dosaženo či dosaženo býti má.

4.6 Spojení

Prvním krokem implementace je navázání spojení a stabilizace datového mostu mezi informačním systémem společnosti a ekonomickým systémem – FlexiBee. Připomínám, že FlexiBee je postaveno na architektuře cloudového uložiště s integrovaným přístupem prostřednictvím RESP-API HTTP příkazů (*requests*). Z pohledu aplikační logiky budou úkony zajišťovat funkce (*metody*) modelu *Flexibee* (modelem je tímto uvažována speciální třída Nette frameworku). Tento model pak musí nutně přebírat instanci třídy *Accounting*, která mimo jiné předává informace o přihlašovacích údajích pro komunikaci se vzdálenými FlexiBee servery.

```
class Flexibee extends BaseModel {

    public $accounting;

    public function __construct($accounting) {
        $this->accounting = $accounting;
    }
```

Kód 1 – Konstrukce *Flexibee*. (Flexibee.php)

Připojení je realizováno prostřednictvím funkce *curl*, která je doplňkem (extension) PHP. Samotné připojení zajišťuje metoda *connect()*.

```
public function connect() {
    $connection = curl_init();

    // Return content as a string from curl_exec
    curl_setopt($connection, CURLOPT_RETURNTRANSFER, TRUE);
    // Follow redirects (compatibility for future changes in FlexiBee)
    curl_setopt($connection, CURLOPT_FOLLOWLOCATION, TRUE);
    // HTTP authentication
    curl_setopt($connection, CURLOPT_HTTPAUTH, TRUE);
    // FlexiBee by default uses Self-Signed certificates
```

```
    curl_setopt($connection, CURLOPT_SSL_VERIFYPEER, FALSE);
    curl_setopt($connection, CURLOPT_SSL_VERIFYHOST, FALSE);
    // LOG IN
    curl_setopt($connection, CURLOPT_USERPWD, $this->accounting['sUser'] . ':' .
        $this->accounting['sPassword']);

    return $connection;
}
```

Kód 2 – Metoda *connect*. (Flexibee.php)

Třída *Flexibee* kromě samotného připojení zajistí také příjem a předání dat (datový most) implementací následujících metod.

Metoda *getRecord* přijímá tři parametry – identifikátor záznamu v rámci ES (*int)*, označení evidence *(string)* a výstupní formát *(string*). Metoda nejprve vytvoří nové připojení ke vzdálenému serveru a následně vygeneruje *url* požadavek a ten příslušnou metodou (*GET*) odešle.

```
public function getRecord($id, $evidence, $format = 'json') {
    $connect = $this->connect();

    // Create request uri.
    curl_setopt($connect, CURLOPT_URL,
            $this->accounting['sHost'] . '/c/' . $this->accounting['sCompany']
            . '/' . $evidence . '/' . $id . '.' . $format);

    // Set http operation.
    curl_setopt($connect, CURLOPT_CUSTOMREQUEST, 'GET');
    // Send request.
    $response = curl_exec($connect);
…
```

Kód 3 – Metoda *getRecord*. (Flexibee.php)

Návratovou hodnotou je jediný záznam dané evidence v určeném formátu. Při komunikaci spojení mohou nastat neočekávané problémy či výjimky, které je nutno před předáním výstupu ošetřit. (Toto ošetření je dále použito také u ostatních metod třídy *Flexibee*).

```
…
if ($response === false)
    throw new FlexibeeException("Cannot connect: " . curl_error($connect),
            FlexibeeException::CANNOT_CONNECT);

$code = curl_getinfo($connect, CURLINFO_HTTP_CODE);

if ($code != 201 && $code != 200 && $code != 400)
    throw new FlexibeeException("HTTP request failed, status: "
            . $code, FlexibeeException::REQUEST_FAILED);

if ($code == 400)
    throw new FlexibeeException("Request failed: "
            . $result->results->result->errors->error,
```

```
            FlexibeeException::BAD_REQUEST, $result);
```

Kód 4 – Ošetření možných výjimek. (Flexibee.php)

Z kódu je zřejmé, že v podstatě mohou nastat v rámci požadavku 3 typy výjimek (které jsou odchytávány). Buďto není vůbec navázáno spojení se serverem, nebo selže *HTTP* požadavek, případně požadavek dorazí v pořádku, ale může být konstruován chybně.

Metoda *get* – na rozdíl od metody *getRecord* vrací všechny záznamy dané evidence vyhovující podmínkám a filtrům nastavených v parametrech funkce, kterými jsou – označení evidence *(string)*, filtry ve formě omezujících podmínek *(string)*, parametrické nastavení výstupu např. detailnost výstupu *(string*), výstupní formát *(string)*. Při sestavování http požadavku je v tomto případě nutno vzít v potaz všechny hodnoty vstupních parametrů což demonstruje následující úryvek kódu.

```
public function get($evidence, $filter = NULL, $param = array('detail' => 'full'),
$format = 'json') {
    $connect = $this->connect();

    isset($filter) ? $urlFilter = '(' . $filter . ')' : $urlFilter = NULL;
    !isset($param['limit']) ? $param['limit'] = 0 : NULL;
    isset($param) ? $urlParam = '?' . http_build_query($param) : $urlParam = array();

    $url = $this->accounting['sHost'] . '/c/'
            . $this->accounting['sCompany'] . '/' . $evidence . '/'
            . rawurlencode($urlFilter) . $urlParam;
…
```

Kód 5 – Metoda *get* sestavení http požadavku. (Flexibee.php)

Zpracování požadavku a ošetření výjimek je pak velmi podobné jako v již popsané metodě *getRecord*. Tato metoda je dále v aplikaci jednou z nejvytíženějších, protože se obvykle využívána k získání „surových dat“ určených pro další zpracování. (Metoda *getRecord* obvykle vrací konkrétní koncový dokument například ve formátu PDF).

Metoda *put* – jak už název napovídá, obsluhuje odesílání dat na vzdálené servery ES. Formát odesílaných dat je vždy *XML*, přičemž ve stejném formátu server vrací také odpověď (v případě, že *HTTP* požadavek je rozpoznán a zpacován). Vstupními parametry metody jsou – označení evidence, do které je záznam odesílán *(string)*, odesílaná data jako pole *(associative array)*, identifikátor záznámu v rámci ES *(int*) jako parametr nepovinný sloužící k editaci exitujícího záznamu. (Další možností identifikace záznamu pro editaci je uvedení identifikátoru jako součást pole s daty). Sestavení požadavku probíhá dle následujícího úryvku kódu.

```
public function put($evidence, $data, $flexiId = null) {
    $connect = $this->connect();

    // Create request uri
    $url = $this->accounting['sHost'] . '/c/'
            . $this->accounting['sCompany'] . '/' . $evidence . '.xml';

    if ($flexiId)
        $url = $this->accounting['sHost'] . '/c/'
            . $this->accounting['sCompany']
            . '/' . $evidence . '/' . $flexiId . '.xml';

    // Set up http request
    curl_setopt($connect, CURLOPT_URL, $url);
    curl_setopt($connect, CURLOPT_HTTPHEADER, array('Accept: application/xml'));
    curl_setopt($connect, CURLOPT_CUSTOMREQUEST, 'PUT');

    // Own method create a new SimpleXML object
    $xml = $this->createXML($data, $evidence);
    curl_setopt($connect, CURLOPT_POSTFIELDS, $xml->asXML());

    $response = curl_exec($connect);
…
```

Kód 6 – Metoda *put*. (Flexibee.php)

Po sestavení url pro odeslání požadavku jsou data předána metodě *CreateXML* která vytvoří nový *SimpleXML* objekt. Tento objekt je naplněn daty, která jsou konvertována z pole na *XML* řetězec prostřednictvím metody *arrayToXml*. (Uvedené metody jsou taktéž součástí třídy *Flexibee*.) Výsledek *HTTP* požadavku, v případě že nenastane některá z výjimek, je zpětně převeden z *XML* na pole a toto dále předáno jako návratová hodnota funkce. Odpověď serveru může vypadat následovně:

```
<winstrom version="1.0">
    <success>true</success>
    <stats>
        <created>0</created>
        <updated>1</updated>
        <deleted>0</deleted>
        <skipped>0</skipped>
        <failed>0</failed>
    </stats>
    <results>
        <result>
            <id>1</id>
            <request-id>1</request-id>
            <ref>/c/sledovanitv_cz_s_r_o___testovaci/adresar/1.xml</ref>
        </result>
    </results>
</winstrom>
```

Kód 7 – Odpověď FlexiBee serveru informující o úspěšné editaci záznamu.

4.7 Synchronizátor

Je-li nastaven způsob a pravidla komunikace mezi servery, pak je možné posunout se dále k popisu objektu synchronizátoru, který je mezistupněm mezi prací s daty v IS a datovým mostem mezi ES. Úlohou synchronizátoru je vytvoření správné instance připojení do ES (získání přístupových dat) a příprava dat k jejich odeslání. Následující diagram zobrazuje tok dat a jejich formát.

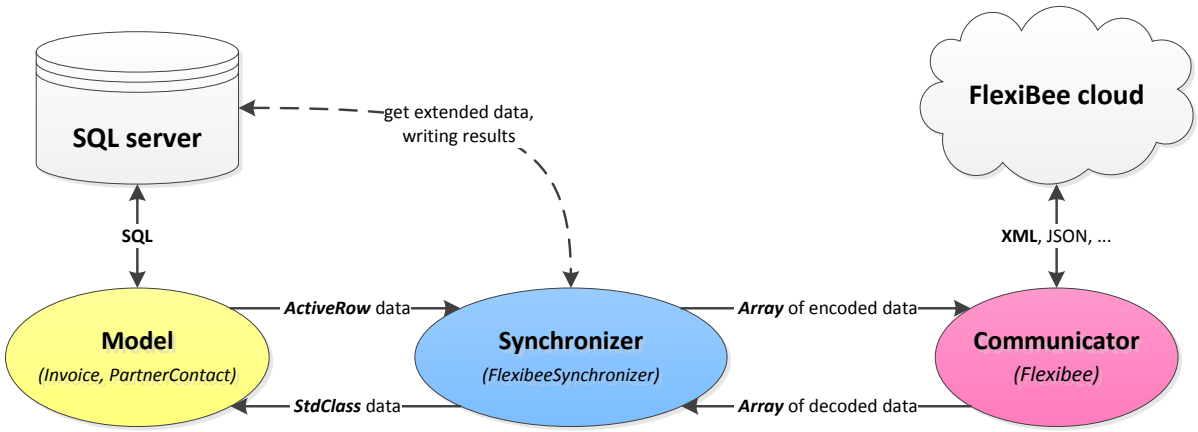


Diagram 13 – Tok a stupně zpracování dat mezi IS a ES. (Zdroj: vlastní tvorba)

Ve stručnosti k diagramu - model či presenter si vytvoří novou instanci synchronizátoru dle objektu, se kterým chce pracovat (kontakty, faktury, …), a tomu předá data a informace (parametry) pro realizaci spojení (např. přihlašovací data k ES). Nová instance pak poskytuje řadu metod, které poskytují potřebné služby. Tyto služby data překládají, doplňují (např. z DB), případně jinak zpracují. Připravená data jsou následně předána třídě *Flexibee* (pro kterou je v diagramu použit pracovní název *Communicator*) a ta již zajišťuje pouze přenos/příjem dat tak, jak byl popsán v předchozí kapitole. V následujících řádcích uvádím několik těchto služeb (metod), které považuji za kritické a které budou v provozu nejvytíženější.

4.7.1 Synchronizátor kontaktů

Synchronizátor kontaktů je realizován prostřednictvím třídy *FlexibeeContact*. V konstruktoru přijímá údaje o ES a zároveň vytváří novou instanci třídy *Flexibee*. Poskytuje následující služby:

Metoda *pull* – získání všech záznamů z FlexiBee evidence „adresar“ s omezením pouze na dodavatele a odběratele. Metoda vrací pole *(array)* přijatých hodnot. Indexy návratového pole

jsou shodné s těmi, které definuje FlexiBee dokumentace a které jsou odlišné od atributů v interní databázi.

```
public function pull() {
    $data = $this->flexibee->get('adresar', "typVztahuK='typVztahu.odberDodav'"
            . "or typVztahuK='typVztahu.odberatel'");
    $object = json_decode($data);
    return $object->winstrom->adresar;
}
```

Kód 8 – Metoda *pull*. (FlexibeeContact.php)

Metoda *push* – obsluhuje odeslání jediného záznamu do FlexiBee pro zápis či editaci.

```
public function push($data) {
    return $this->flexibee->put('adresar', $data);
}
```

Kód 9 – Metoda *push*. (FlexibeeContact.php)

Metoda ***synchronize*** – přijímá všechny dostupné kontakty z interní databáze i ES a vzájemně synchronizuje změny. Zdrojový kód řeší řadu úskalí a uvedení stěžejních úseků kódu by ani z části nebylo tak výmluvné (kód je obsáhlý) jako použití následujícího diagramu, který práci služby v několika procesech shrnuje.

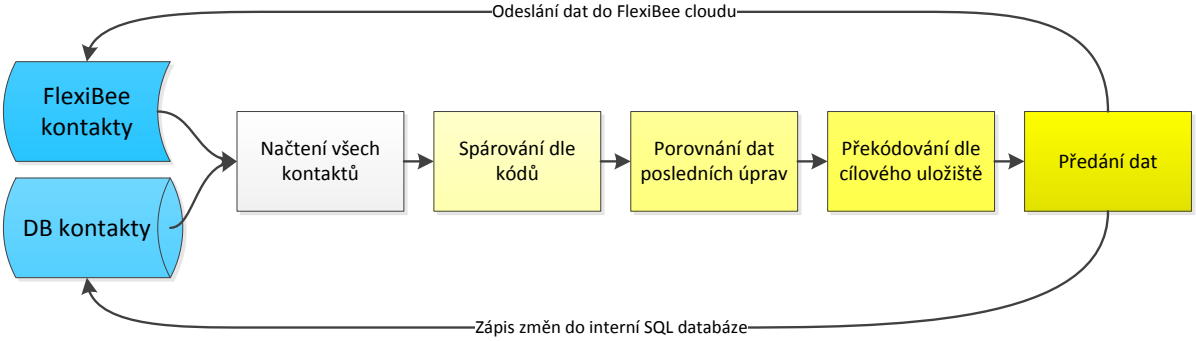


Diagram 14 – Jednotlivé procesy synchronizace kontaktů. (Zdroj: vlastní tvorba)

Existující kontakty jsou v obou případech uložiště aktualizovány, neexistující kontakt ve FlexiBee je vytvořen a neexistující kontakt v IS je ignorován.

Metoda *updateDB* – načte všechny dostupné kontakty z adresáře FlexiBee vyhovující podmínky dle metody *pull* a tyto na základě kódů kontaktů aktualizuje v interní databázi. Neexistující kontakty nejsou (stejně jako v metodě *synchronize*) do DB uloženy. (Kontakty, se kterými IS pracuje, jsou primárně v IS také vytvořeny a ostatní nemají být duplikovány.)

4.7.2 Fakturační synchronizátor

Synchronizátor faktur je implementován třídou *FlexibeeSynchronizer*. Kromě práce s objekty samotných faktur pracuje také s položkami faktur, které jsou samostatnými objekty. Instanci třídy je možno získat podobným způsobem jako u kontaktů, tedy předáním *Accounting*. Synchronizaci fakturace zajišťují následující metody.

Metoda *pull* – prosté získání všech faktur z uložiště ekonomického systému Flexibee.

Metoda *pullByPartner* – parametrem je předán objekt *Partners*, dále program voláním statické metody *getBy* objektu *PartnerContact* extrahuje kód firmy, který použije jako filtr pro získání záznamů z požadované evidence *(faktura-vydana)*.

```
public function pullByPartner($partner = null) {
    $contact = PartnerContact::getBy(array('partner' => (string) $partner));
    $data = $this->flexibee->get('faktura-vydana', "firma='code:".$contact['code']."'");
    $object = json_decode($data);
    return $object->winstrom->{'faktura-vydana'};
}
```

Kód 10 – Metoda *pullByPartner*. (FlexibeeSynchronizer.php)

Metoda *pullById* – získá jako parametr FlexiBee identifikátor a tento použije pro získání dat jediného záznamu dané evidence.

```
public function pullById($flexiInvoiceId) {
    $data = $this->flexibee->get('faktura-vydana/' . $flexiInvoiceId);
    $object = json_decode($data);
    return $object->winstrom->{'faktura-vydana'};
}
```

Kód 11 – Metoda *pullById*. (FlexibeeSynchronizer.php)

Metoda *pullItems* – získá položky dané faktury. Parametrem je předán pouze identifikátor faktury. Zajímavý je zde způsob postupného skládání *HTTP* požadavku, který je na podobném principu možno využívat u většiny dalších evidencí.

```
public function pullItems($flexiInvoiceId) {
    $data = $this->flexibee->get('faktura-vydana/' . $flexiInvoiceId .
'/polozkyDokladu');
    $object = json_decode($data);
    return $object->winstrom->{'faktura-vydana-polozka'};
}
```

Kód 12 – Metoda *pullItems*. (FlexibeeSynchronizer.php)

Metoda *getPdf* – umí díky již popsaným metodám třídy *Flexibee* velmi elegantním způsobem vrátit na základě FlexiBee identifikátoru fakturu ve formátu *PDF*, která může být dále vytištěna či zaslána klientovi.

```
public function getPdf($flexiInvoiceId) {
    return $this->flexibee->getRecord($flexiInvoiceId, 'faktura-vydana', 'pdf');
}
```

Kód 13 – Metoda *getPdf*. (FlexibeeSynchronizer.php)

Metody *convertInvoiceData* a *convertContactData* – zajišťují konvertování dat pro určené uložiště. Každé uložiště totiž používá jiné názvy atributů v rámci databázové entity a jiné datové typy a formáty. Parametry jsou vždy pole dat a určení databáze, pro kterou mají být data připravena.

Metoda *push* – je stěžejní službou synchronizátoru a zajišťuje přípravu dat pro odeslání do ES. Funkce musí kromě faktury jako takové připravit také související položky faktury, které nelze odeslat samostatně a musí být (z důvodu zachování jednoznačné integrity) odeslány společně v rámci jediného *http requestu*. Přijatým parametrem metody je objekt *Invoice* a nepovinným parametrem položky faktury pole objektů *InvoiceItems*. V případě, že položky faktury nejsou předány, pak funkce načte všechny příslušné DB záznamy.

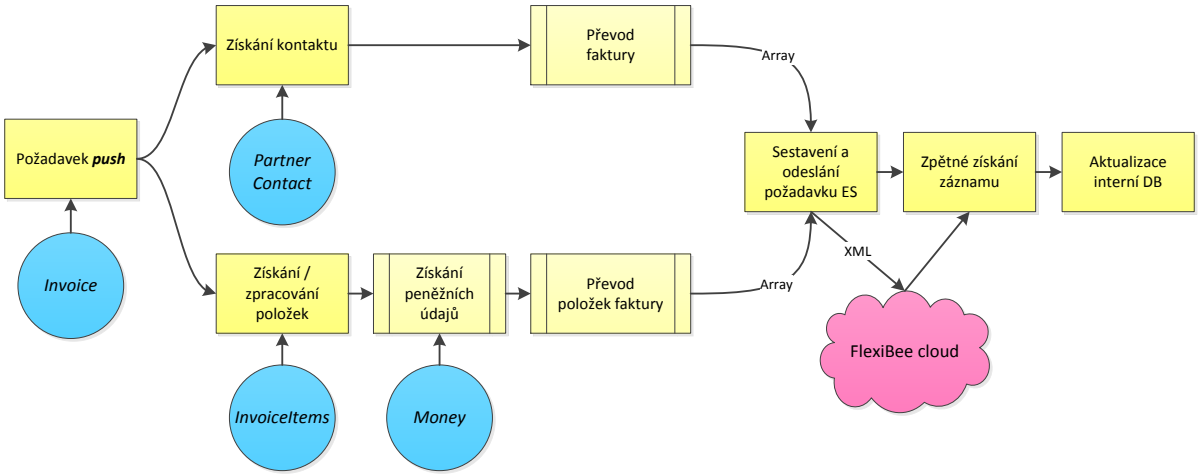


Diagram 15 – Jednotlivé procesy metody *push*. (Zdroj: vlastní tvorba)

Z diagramu je patrné, že metody zpracuje fakturu ve dvou procesech, v prvním připraví data samotné fakturu a následně data všech položek. Zkonvertovaná data jsou předána komunikátoru, který tato odešle do FlexiBee cloudu. Po odeslání jsou data (zaktualizovaná)

znovu stažena a uložena v databázi. (Odesláním nové faktury přidělí ES variabilní symbol a další údaje, které je nutno ihned přenést zpět do interní databáze.)

Metoda *synchronize* – pátrá po změnách ve fakturačních záznamech obou databází a neaktuální záznamy doplňuje aktuálními daty. Synchronizace probíhá v následujících krocích:

1. **Načtení dat**
    a. Načte všechny záznamy z lokální databáze *(getList)* a zároveň všechny požadované záznamy z FlexiBee *(pull)*.
    b. Záznamy páruje a v případě, že záznamy mají odlišné datum poslední úpravy, aktualizuje ten se starším datem.
    c. Program ukládá informace o záznamech, které již zpracoval.
2. **Nové ES** záznamy
    a. Nezpracované záznamy z FlexiBee jsou opětovně procházeny a program zjišťuje existenci partnera v DB.
    b. Najde-li partnera, pak záznam ukládá do DB jako novou vydanou fakturu.
    c. Po uložení nového záznamu je volána funkce *synchronizeItems*.
3. **Nové IS** záznamy
    a. Nezpracované záznamy z interní databáze jsou jednotlivě odesílány do Flexibee. (Tato funkce je implementována ale nyní pozastavena, protože před odesláním faktury do ES musí být provedena manuální kontrola.)

Návratovou hodnotou je písemný protokol o změnách. (Výpis voláním *getProtocol*).

Metoda *synchronizeItems* – jako parametr přijímá objekt *Invoice* přičemž dále pracuje na úrovni jednotlivých položek faktury, které jsou v rámci faktury synchronizovány.

```php
// Create new Money record
$money = new MoneyWithoutVat(1500, 21, 'CZK');
$money->save();

// Get money object
$invoiceItem = InvoiceItem::getById(15);
$money = $invoiceItem->getMoney();

// Update Money with DB auto-saving
$money = array(
    'withoutVat' => 1600,
    'vatPercent' => 21,
    'currency' => 'CZK'
);
```

Kód 14 – Metoda *SynchronizeItems*. (FlexibeeSynchronizer.php)

4.8 Aplikační modely

Jedná se o jednotlivé modely (ve významu dle *Nette* názvosloví), které jsou aplikačním obrazem objektu reálného světa a implementují jeho vlastnosti. V souvislosti s touto prací se jedná o třídy:

1. ***Accounting***
2. ***PartnerContact***
3. ***Invoice***
4. ***InvoiceItem***

Všechny uvedené objekty jsou potomky třídy *BaseEntity*, která je určena pro provádění služeb nad databázovou entitou, kterou od potomka obdrží. Tito potomci dědí metody svého rodiče a dále je rozšiřují. Jinak řečeno, každá z uvedených tříd zastřešuje práci s jednou relací SQL databáze. Popis všech uvedených metod, jejich vlastností a možností by byl velmi rozsáhlý, proto se budu snažit uvést na příkladech jejich použití a praktické využití.

4.8.1 Model *Accounting*

Kromě standardních funkcí pro práci s entitou Accounting zajišťuje získání instance patřičného synchronizátoru na základě nastavení, které databázový záznam udržuje. Příklad volání metod demonstruje následující ukázka kódu.

```
// Get accounting record from DB
$accounting = Accounting::getById(1);

// Get current invoice synchronizer (FlexibeeSynchronizer)
$accounting->getSynchronizer();

// Get current contact synchronizer (FlexibeeContact)
$accounting->getContactSynchronizer();
```

Kód 15 – Ukázka práce s objektem *Accounting*.

Uvedené služby nejčastěji využívají objekty, které v popisu budou následovat - *PartnerContact* a *Invoice* kde bude využití tohoto modelu zřejmé.

4.8.2 Model *Partner Contact*

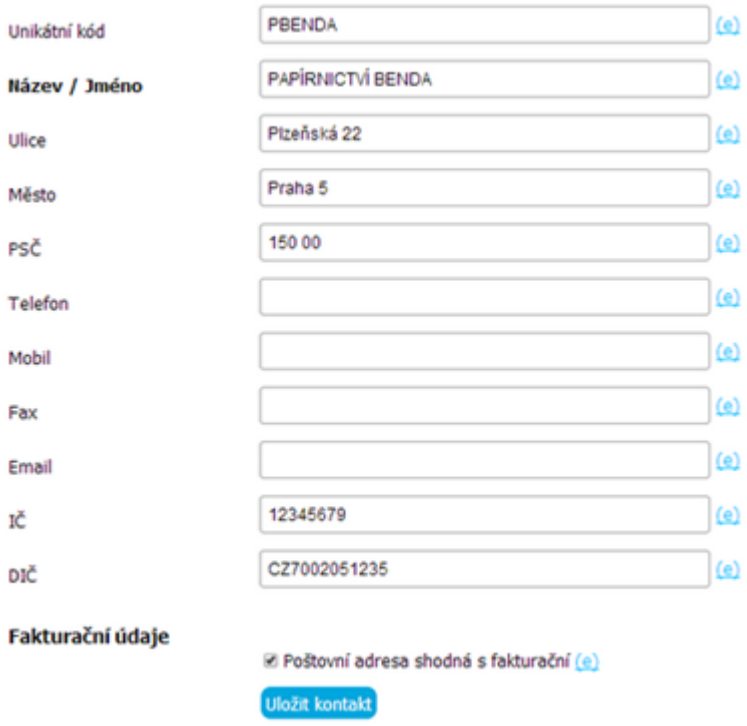
Sdružuje metody zajišťující služby ukládání, získávání a práce s kontakty partnerů (připomínám, že kontakt je samostatná databázová entita, jejíž název je shodný s názvem

aplikačního modelu). Následující ukázka kódu demonstruje možnosti výběru partnera z databáze (návratovou hodnotou je Nette objekt *ActiveRow*).

```php
// Get contact by ID
$contact = PartnerContact::getById(1);

// Get contact by code
$contact = PartnerContact::getByCode('MACHALA');

// Get contact by Partner (class Partner)
$partner = Partner::getById(1);
$contact = PartnerContact::getByPartner($partner);

// Get list of contacts by Accounting
$accounting = Accounting::getById(1);
$contacts = PartnerContact::getListByAccounting((string) $accounting);
```

Kód 16 – Příkladový kód práce s kontakty (třída *ParnetContact*).

Model *PartnerContact* využívá například komponenta, pro editaci kontaktů v klientském rozhraní. Komponenta zajišťuje front-endovou část aplikace a potřebná data od modelu přijímá a zasílá zpět.



Obrázek 6 – Ukázka formuláře pro přidání/editaci kontaktu.

Připojením synchronizátoru získáváme velmi silný nástroj pro práci s kontakty. Následující příklad získá z databáze kontakt, vytvoří novou instanci synchronizátoru a demonstruje možnosti jeho dalšího použití.

```php
// Create instance of synchronizator by Accounting
$accounting = Accounting::getById(1);
$synchronizator = new FlexibeeContact($accounting);

$synchronizator->pull(); // Get all ES contacts
$synchronizator->push($data); // Save contact to ES
$synchronizator->updateDB(); // Update DB according to ES data
$synchronizator->synchronize(); // Synchronize both databases
```

Kód 17 – Kombinace využití modelu a synchronizátoru kontaktů.

Metoda *updateDB* bude použita především při prvním inicializaci aplikace a zajistí přenos a spárování všech kontaktů ES. Při další práci se předpokládá, že budou kontaktní data upravovat sami zákazníci v IS a změny tak budou přenášeny ve směru IS → ES.

4.8.3 Model *Invoice* a *Items*

Aplikační model zpracovávající fakturaci disponuje širokou řadou metod. Znovu se zaměřím pouze na ukázky použití metod. Následující ukázka kódu demonstruje základní operace získání faktury z interní databáze, vytvoření a uložení nové faktury včetně přidání položek.

```php
// Create new Invoice instance
$invoice = new Invoice;

// Exists invoice for partner in specified period?
$partner = Partner::getById(5);
$invoice->checkExists($partner, 2014, 03); // Return bool

// Create new invoice for partner
$invoice->createForPartner($partner, 2014, 03);
// And add items
$money = new MoneyWithoutVat(500, 21, 'CZK'); // Create Money
$item = $invoice->addItem($money, 2, 'Fakturace služeb 2014-03', 'sales_services');

// Do you want to change an item?
$invoice->updateItem((string) $item, $money, 3, 'Fakturace zboží 2014-03',
'sales_goods');

// Get all items from invoice
$invoice->getItems();

// Get partner invoice in specified month
$invoice->getByPartner($partner, 2014, 3);
```

Kód 18 – Ukázka vytvoření nové faktury a přidání položek.

Postupně je inicializována nová faktura, načten partner a ověřeno, že pro dané období již pro partnera není vyúčtování vytvořeno. Následně je nové vyučtování uloženo, vytvořen nový peněžní objekt (*Money*) a na základě něj pak nová položka faktury, která je k faktuře přiřazena včetně ukázky editace. Závěrem program získá všechny položky faktury, případě vyúčtování pro daného partnera v určeném období.

Práce s třídou dále dostává nový rozměr zapojením fakturačního synchronizátoru. Výsledky předešlých operací tak můžeme snadno směrovat do ekonomického systému FlexiBee.

```php
$synchronizer = new FlexibeeSynchronizer($accounting);

// Get PDF invoice document
$synchronizer->getPdf($invoice['flexibee']);

$synchronizer->pull(); // Get all ES invoices
$synchronizer->pullByPartner(); // Get invoices for partner
$synchronizer->pullById($invoice['flexibee']); // Get invoice by ES identificator
$synchronizer->pullItems($invoice['flexibee']) // Get ES invoice items
$synchronizer->push($invoice); // Send invoice to ES

$synchronizer->synchronize(); // Synchronize invoices in both DB
$synchronizer->synchronizeItems($invoice); // Synchronize items by invoice
```

Kód 19 – Použití třídy invoice v kombinaci se synchronizátorem.

Ukázka demonstruje získání *PDF* faktury, všech faktur, všech faktur pro vybraného partnera, konkrétní fakturu na základě jejího identifikátoru, všechny položky dotyčné faktury, odeslání fakturačních údajů do FlexiBee. Poslední dva řádky jsou věnovány kompletní synchronizaci a synchronizaci položek vybrané faktury.

Data přijatá z FlexiBee aplikace také umí zpracovat. Následující příklad demonstruje postup získání určité faktury z ES, vytvoření nového kontaktu, vytvoření nové faktury, nalezení a přidělení faktury partnerovi a synchronizaci položek faktury.

```php
$invoiceData = $synchronizer->pullById(1278);

// Create new invoice contact from data
$contactData = array('name' => $invoiceData['jmeno'], ...);

// Save invoice to DB
$invoice->create($invoiceData);

// Attach to partner
$invoice->attachToPartner($partner, 2014, 1);

// Synchronize items (save into DB)
$synchronizer->synchronizeItems($invoice);
```

Kód 20 – Ukázka zpracování přijatých dat z ES.

Program získá FlexiBee fakturu dle identifikátoru a vytvoří (přeloží) pole fakturačních (kontaktních) údajů. Na základě fakturačních dat je vytvořena nová faktura v IS a ta přidělena již dříve načtenému partnerovi. Závěrem je provedena synchronizace fakturačních položek.

4.9 Ekonomické zhodnocení

Téměř zanedbatelnou částkou jsou licenční (fixní) náklady ekonomického systému Flexibee, který společnost díky multilicenci v rámci Jihomoravského inovačního centra získává v závislosti na využitých službách a modulech za částku od 5 do 10 tisíc Kč ročně. V souvislosti s provozem software nevznikají žádné dodatečné náklady (veškerá infrastruktura je zajištěna poskytovatelem – FlexiBee Systems).

Náklady analýzy a vývoje – představují náklady, které společnosti vznikají v souvislosti s realizací tohoto projektu. Tyto rozděluji do dvou skupin:

1. **Externí** – náklady, které firma vynakládá za externího dodavatele řešení, kterým jsem v tomto případě já sám. Konkrétní částka může být dosažena výpočtem součinu nákladů mojí hodinové práce a počtu odpracovaných hodin plus ostatní náklady (technické vybavení, opotřebení zařízení) a náklady obětované příležitosti. Abychom se dále pohybovali v konkrétních číslech, pak hodnotím svoji práci na 15 000 Kč.
2. **Interní** – výhradně čas pracovníků společnosti, kteří mi poskytovali informace a konzultace, což samozřejmě zabralo čas, který by jinak věnovali své vlastní práci. Tyto interní náklady jsem vyčíslil na 2 000 Kč.

Náklady původního řešení – původní řešení, tak jak bylo popsáno, zaměstnalo účetní pracovnici na několik dnů. Tato pracovnice je zaměstnána na trvalý poměr a třetinu své pracovní doby musela věnovat úkonům, které tato práce řešila. Pracovnice i po nasazení řešení bude práci vykonávat, ale bude se moci věnovat jiným (důležitějším) aktivitám. Náklady původního stavu jsem vyčíslil na 75 000 Kč ročně.

Náklady nového řešení – jsou spojeny se zajištěním fungování nového řešení. Zanedbám zde částku provozu a údržby zařízení (její přesné vyčíslení by bylo příliš nepřesné a pohybovalo by se v nízkých hodnotách) a zaměřím výhradně na náklady práce, kdy účetní stráví stejnými úkony již jen 5 % své pracovní doby, což ročně ohodnocuji na částku 15 000 Kč.

$$\boldsymbol{TC\ implementace} = 15\,000\ Kč\ (externí) + 2\,000\ Kč\ (interní) = \mathbf{17\,000}\ Kč$$

$$Roční\ náklady\ \boldsymbol{původního}\ řešení = 8\,000\ Kč\ (license) + 75\,000\ Kč\ (práce) = \mathbf{83\,000}\ Kč$$

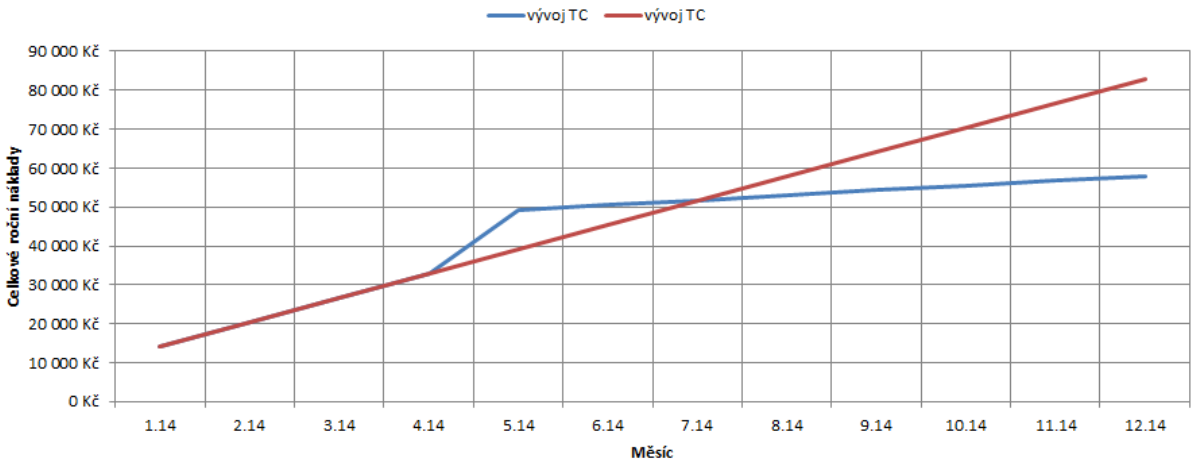
$$Roční\ náklady\ \boldsymbol{nového}\ řešení = 8\,000\ Kč\ (license) + 15\,000\ Kč\ (práce) = \mathbf{23\,000}\ Kč$$

Doba návratnosti i investice – následující tabulka vyjadřuje postupný vývoj nákladů spojených s předmětnými činnostmi od počátku roku 2014. (Je fakturována licence ES). Původní řešení počítá s neměnnými variabilními náklady po celé období. Nové řešení předpokládá fixní náklady v květnu 2014 a související snížení variabilních nákladů.

	Původní řešení			Nové řešení		
Měsíc	VC	FC	vývoj TC	VC	FC	vývoj TC
1.14	6 250 Kč	8 000 Kč	14 250 Kč	6 250 Kč	8 000 Kč	14 250 Kč
2.14	6 250 Kč	0 Kč	20 500 Kč	6 250 Kč	0 Kč	20 500 Kč
3.14	6 250 Kč	0 Kč	26 750 Kč	6 250 Kč	0 Kč	26 750 Kč
4.14	6 250 Kč	0 Kč	33 000 Kč	6 250 Kč	0 Kč	33 000 Kč
5.14	6 250 Kč	0 Kč	39 250 Kč	1 250 Kč	15 000 Kč	49 250 Kč
6.14	6 250 Kč	0 Kč	45 500 Kč	1 250 Kč	0 Kč	50 500 Kč
7.14	6 250 Kč	0 Kč	51 750 Kč	1 250 Kč	0 Kč	51 750 Kč
8.14	6 250 Kč	0 Kč	58 000 Kč	1 250 Kč	0 Kč	53 000 Kč
9.14	6 250 Kč	0 Kč	64 250 Kč	1 250 Kč	0 Kč	54 250 Kč
10.14	6 250 Kč	0 Kč	70 500 Kč	1 250 Kč	0 Kč	55 500 Kč
11.14	6 250 Kč	0 Kč	76 750 Kč	1 250 Kč	0 Kč	56 750 Kč
12.14	6 250 Kč	0 Kč	83 000 Kč	1 250 Kč	0 Kč	58 000 Kč

Tabulka 21 – Vývoj souvisejících nákladů roce 2014.

Z tabulky (a následující graf to potvrzuje) vyplývá, že náklady na nové řešení se s náklady původního řešení vyrovnají v červenci 2014 a poté do konce roku ušetří 25 000 Kč.



Graf 5 – Vývoj souvisejících ročních nákladů.

Závěrem skutečné náklady jsou nižší než uvedené, jelikož za svou práci neinkasuji žádné prostředky ale jen zkušenosti, které jsou pravděpodobně mnohem hodnotnější. Společnost zároveň u tohoto výstupu nezůstává, ale hodlá jej dále rozvíjet (automatizace zpracovávání plateb, řízení pohledávek, nasazení CRM).

4.10 Strategie budoucího rozvoje IS

Při zpětném pohledu na informační systém jako celek ve spojitosti s provedenou analýzou HOS 8 se na základě mé práce zlepšila situace minimálně ve dvou zkoumaných oblastech – software a dataware. V případě software se jedná o vytvoření nových funkcí IS, které uživatelé budou využívat ke své práci. V případě pracovnice, která zajišťuje účetní agentu, se navíc jedná o funkce, které k práci potřebuje zcela nezbytně. V oblasti dataware pak integrace IS a ES přináší lepší přehled a správu účetních dat, lepší vymezení zodpovědností a zároveň systém poskytuje i nová data, která uživatelům pomáhají v rozhodování.

V otázce budoucího rozvoje IS navrhuji společnosti několik opatření, které by, vzhledem k charakteru její činnosti, mohly přinést pozitivní přidanou hodnotu:

1. Tvorba a sjednocení programově-dokumentační metodiky alespoň formou výčtu funkcí a způsobu užití jednotlivých komponent, tříd a metod. V případě dalšího rozvoje společnosti a vzniku potřeby přijmout další vývojáře s velkou pravděpodobností (díky neexistující nebo nejednotné dokumentaci) vzniknou problémy a časová náročnost zaškolování.

2. Specifikace postupů a vytvoření (základní) směrnice pro postupy v případě nestandardních či havarijních situaci a stanovení bezpečnostní politiky a zajištění aktualizace, doplňování a rozvoj těchto dokumentů.

3. Zlepšení správy a managementu klientských dat. (Společnost v dohledné době plánuje nasazení CRM systému.)

4. Stanovení cílů a metrik budoucího rozvoje IS. Toto doposud probíhá pouze jako vize vedoucích pracovníků bez vytváření hmatatelných podkladů, na základě kterých by mohlo být vyhodnocováno dosahování stanovených milníků rozvoje.

ZÁVĚR

V práci byl nejprve identifikován záměr, stanoven cíl a posouzena jeho reálnost. V teoretické části byly položeny základy budoucí analýzy, návrhu řešení a jeho implementace, popsány principy informačních systémů a informačního managementu, představeny konkrétní technologie – technologické nástroje, metody a postupy.

Analytická část systematicky nejprve představila společnost a její interní informační systém, identifikovala organizační strukturu, štáby a jejich propojení, zainteresované osoby a jejich vazby k IS. Informační systém byl podroben analýze metodou HOS 8 – vyvozeny konkrétní nedostatky a analyzovány požadované změny.

V praktické části bylo nejprve nalezeno a popsáno řešení a způsoby integrace. Stěžejní část se věnovala popisu samotné implementace funkcionalit – aplikační logika, způsoby použití a nasazení. Závěr části se věnuje ekonomickým aspektům – vyčíslení nákladů, demonstrace doby návratnosti a vyhodnocení ekonomického přínosu.

Vyladění programu proběhlo v kooperaci s firemními vývojáři, kteří také zajistili otestování funkčnosti na reálných datech a nasazení do ostrého provozu (postupnou strategií). Od května roku 2014 výstup této práce zpracovává fakturační podklady více než dvou stovek klientů společnosti. Tímto považuji cíl práce za splněný.

Kromě automatického zpracovávání vyúčtování chce společnost implementovat také zpracovávání plateb (z různých platebních nástrojů a systémů). Zároveň na základě rychle rostoucího počtu klientů firma v blízké budoucnosti předpokládá zavedení CRM systému. Podnikatelský záměr, který se podařilo úspěšně realizovat v České republice je nyní (jaro 2014) přenášen také na Slovensko a zástupci společnosti plánují postupnou expanzi i na další zahraniční trhy.

SEZNAM POUŽITÉ LITERATURY

Monografické zdroje

BASL, J. a BLAŽÍČEK, R., 2008. Podnikové informační systémy : Podnik v informační společnosti. 2. vyd. Praha: Grada, 283 s. ISBN 978-80-247-2279-5.

DOSTÁL, P. a RAIS, K. a SOJKA, Z., 2005. Pokročilé metody manažerského rozhodování. 1. vyd. Praha: Grada, 168 s. ISBN 80-247-1338-1.

DOVRTĚL, J., c2005. Vybrané aspekty efektivnosti informačních systémů. [s.l.], 30 s. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Dizertační práce. ISBN 80-214-2891-0.

KOFLER, M., 2007. Mistrovství v MySQL 5. 1. vydání. Brno: Computer Press. 808 s. ISBN 978-80-251-1502-2.

KOCH, M., et al., 2010. Management informačních systémů. 3. přepracované. Brno: Akademické nakladatelství CERM, 171 s. ISBN 978-802-1441-576.

MOLNÁR, Z., 2000. Efektivnost informačních systémů. 1. vyd. Praha: Grada, 144 s. ISBN 80-7169-410-X.

MOLNÁR, Z., 1992. Moderní metody řízení informačních systémů. Praha: Grada, 347 s. ISBN 80-85623-07-2.

SODOMKA, P. a KLČOVÁ, H., 2010. Informační systémy v podnikové praxi. 2. vyd. Brno: Computer Press, 504 s. ISBN 978-80-251-2878-7.

SODOMKA, P., 2006. Informační systémy v podnikové praxi. 1. vyd. Brno: Computer Press, a.s., 351 s. ISBN 80-251-1200-4.

TVRDÍKOVÁ, M., 2008. Aplikace moderních informačních technologií v řízení firmy: Nástroje ke zvyšování kvality informačních systémů. První vydání. Praha: Grada Publishing, a.s., 176 s. ISBN 978-80-247-2728-8.

Elektronické zdroje

FlexiBee Systems s.r.o., 2014. *Internetové ekonomické systémy FlexiBee* [online]. [cit. 2014-03-15]. Dostupné z: http://www.flexibee.eu

MALÝ, M., 2009. *REST: architektura pro webové API*. Zdroják.cz [online]. 2009-08-03 [cit. 2014-03-15]. ISSN 1803-5620. Dostupné z: http://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

NETTE FOUNDATION, 2014. Nette Framework [online]. 2014 [cit. 2014-03-20]. Dostupné z: http://www.nette.org

BOREK, B., 2009. *Úvod do architektury MVC*. Zdroják.cz [online]. 2009-05-07 [cit. 2014-03-25]. ISSN 1803-5620. Dostupné z: http://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/

THE APACHE SOFTWARE FOUNDATION, 2014. *About the Apache HTTP Server Project* [online]. 2012 [cit. 2014-05-19]. Dostupné z: http://httpd.apache.org/ about_apache.html

THE PHP GROUP, 2012. *PHP general information* [online]. 2012 [cit. 2014-05-01]. Dostupné z: http://cz2.php.net/manual/en/faq.general.php

THE WORLD WIDE WEB CONSORTIUM, 2014. *Planet HTML5* [online]. 2014 [cit. 2014-05-01]. Dostupné z: http://www.w3.org/html/planet/

SEZNAM OBJEKTŮ

Obrázky

Diagramy

Grafy

Zdrojové kódy

Tabulky

Přílohy

Kompletní zdrojové kódy, struktura databáze a další související materiály jsou součástí přiloženého CD.

PŘILOHY

Příloha 1 - FlexiBee XML ukázka faktury.

```
<winstrom version="1.0">
    <adresar update="ignore">
        <kod>WINSTROM</kod>
        <id>code:WINSTROM</id>
        <!--  Název (řetězec) - max. délka: 255  -->
        <nazev>WinStrom s.r.o.</nazev>
        <!--  Ulice (řetězec) - max. délka: 255  -->
        <ulice>Lochotínská 18</ulice>
        <!--  Město (řetězec) - max. délka: 255  -->
        <mesto>Plzeň</mesto>
        <!--  PSČ (řetězec) - max. délka: 255  -->
        <psc>301 00</psc>
        <!--  IČ (řetězec) - max. délka: 20  -->
        <ic>28019920</ic>
        <!--  DIČ (řetězec) - max. délka: 20  -->
        <dic>CZ28019920</dic>
        <!--  Stát (objekt) - max. délka: 3  -->
        <stat>code:CZ</stat>
    </adresar>
    <!--  Doklady faktur  -->
    <faktura-vydana>
        <!--  Variabilní symbol (řetězec) -  -->
        <varSym>20080004</varSym>
        <!--  Vystaveno (datum) -  -->
        <datVyst>2009-01-03+01:00</datVyst>
        <!--  Popis (řetězec) - max. délka: 255  -->
        <popis/>
        <!--  Typ faktury (objekt) - max. délka: 20  -->
        <typDokl>code:FAKTURA</typDokl>
        <!--  Předpis zaúčtování (objekt) - max. délka: 20  -->
        <typUcOp>code:TRŽBA SLUŽBY</typUcOp>
        <!--  Firma (objekt) - max. délka: 20  -->
        <firma>code:WINSTROM</firma>
        <polozkyFaktury>
            <!--  Položky faktur  -->
            <faktura-vydana-polozka>
                <!--  Zkratka (řetězec) - max. délka: 20  -->
                <kod>STROJ SD</kod>
                <!--  Název (řetězec) - max. délka: 255  -->
                <nazev>Psací stroj Super elektrický s pamětí a displayem</nazev>
                <!--...-->
                <typPolozkyK>typPolozky.obecny</typPolozkyK>
                <!--  Množství (destinné číslo) - počet číslic: 19  -->
                <mnozMj>100.0</mnozMj>
                <!--...-->
                <typCenyDphK>typCeny.bezDph</typCenyDphK>
                <!--...-->
                <typSzbDphK>typSzbDph.dphZakl</typSzbDphK>
                <!--  DPH [%] (destinné číslo) - počet číslic: 6  -->
                <szbDph>19.0</szbDph>
                <!--  Cena za MJ (destinné číslo) - počet číslic: 19  -->
                <cenaMj>15200.0</cenaMj>
            </faktura-vydana-polozka>
        </polozkyFaktury>
    </faktura-vydana>
</winstrom>
```

Příloha 2 - FlexiBee XML ukázka kontaktu.

```
<winstrom version="1.0">
    <!--  Adresář  -->
    <adresar>
        <!--  Zkratka (řetězec) - max. délka: 20  -->
        <kod>WINSTROM</kod>
        <!--  Název (řetězec) - max. délka: 255  -->
        <nazev>WinStrom s.r.o.</nazev>
        <!--  Poznámka (řetězec) -  -->
        <poznam/>
        <!--  Popis (řetězec) - max. délka: 255  -->
        <popis/>
        <!--  Ulice (řetězec) - max. délka: 255  -->
        <ulice>Lochotínská 18</ulice>
        <!--  Město (řetězec) - max. délka: 255  -->
        <mesto>Plzeň</mesto>
        <!--  PSČ (řetězec) - max. délka: 255  -->
        <psc>301 00</psc>
        <!--  Telefon (řetězec) - max. délka: 255  -->
        <tel/>
        <!--  Mobil (řetězec) - max. délka: 255  -->
        <mobil/>
        <!--  Fax (řetězec) - max. délka: 255  -->
        <fax/>
        <!--  E-mail (řetězec) - max. délka: 255  -->
        <email>info@winstrom.cz</email>
        <!--  WWW (řetězec) - max. délka: 255  -->
        <www>www.winstrom.cz</www>
        <!--  Stát (objekt) - max. délka: 3  -->
        <stat>code:CZ</stat>
        <!--  IČ (řetězec) - max. délka: 20  -->
        <ic>28019920</ic>
        <!--  DIČ (řetězec) - max. délka: 20  -->
        <dic>CZ28019920</dic>
        <!--  IČ DPH (řetězec) - max. délka: 20  -->
        <vatId/>
        <!--  Plátce DPH (logická hodnota) -  -->
        <platceDph>true</platceDph>
        <!--
         Typ vztahu
              Odběr./Dodav. - typVztahu.odberDodav
              Odběratel - typVztahu.odberatel
              Dodavatel - typVztahu.dodavatel
        -->
        <typVztahuK>typVztahu.odberDodav</typVztahuK>
    </adresar>
</winstrom>
```

Příloha 3 - PDF náhled vydané faktury.

Faktura - daňový doklad

VF1-0002/2008

Dodavatel:
sledovanitv.cz s.r.o. - Testovaci
Holandská 31
100 00 Praha 10
Česká republika
IČ: 21212121
DIČ: CZ21212121

Telefon: 25896544
Fax: 25845634
Mobil:
E-mail:
WWW:

Banka: Komerční banka, a.s.
Bankovní účet: **123456789 / 0100**
IBAN:
BIC:
Var. sym.: **20080002**
Konst. sym.: **0008**
Spec. sym.:

Odběratel - sídlo: **PAPÍRNICTVÍ BENDA**
Plzeňská 22
150 00 Praha 5

IČ: 12345679, DIČ: CZ7002051235

Poštovní adresa:
PAPÍRNICTVÍ BENDA
Plzeňská 22
150 00 Praha 5

Místo určení:

Číslo smlouvy:
Objednávka:
Zakázka: 111888

Forma úhrady: Převodem
Způsob dopravy:

Vystaveno: 10.04.2014
Datum splatnosti: **30.04.2014**
Datum uskutečnění zdanitelného plnění: 30.04.2014

Označení dodávky / Označení	Množství	MJ	Cena za MJ	Sazba DPH	Základ [Kč]	Celkem [Kč]
Objednávka: 12345 ze dne 02.01.2005, zakázka: 111888						
Text:						
Skartovačka papíru 5list.	1,00	KS	2 500,00	19,00	2 500,00	2 975,00
DRTICPAP025						

Rekapitulace DPH v Kč

Základ 0%	0,00	DPH 0%	0,00
Základ 15%	0,00	DPH 15%	0,00
Základ 19%	2 500,00	DPH 19%	475,00
Základ 21%	0,00	DPH 21%	0,00
Celkem	2 500,00		475,00

Celkem k úhradě	**2 975,00**
Zálohy	**0,00**
Zbývá uhradit [Kč]	**2 975,00**

Registrace:
zapsano u KS v Praze, oddíl B vložka 89512

Razítko a podpis

Vytištěno systémem FlexiBee. Stránka 1